UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

ASPECTS IDENTIFICATION IN BUSINESS PROCESS THROUGH PROCESS MINING

Bruna Christina Pinto Brandão

Orientadores

Flávia Maria Santoro

Leonardo Guerreiro Azevedo

RIO DE JANEIRO, RJ – BRASIL

SETEMBRO DE 2015

ASPECTS IDENTIFICATION IN BUSINESS PROCESS THROUGH PROCESS MINING

Bruna Christina Pinto Brandão

DISSERTAÇÃO APRESENTADA COMO REQUISITO PARCIAL PARA OBTENÇÃO DO TÍTULO DE MESTRE PELO PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA DA UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO (UNIRIO), APROVADA PELA COMISSÃO ABAIXO ASSINADA.

Aprovada por

[illegible]

Flávia Maria Santoro, D.Sc. – UNIRIO

[illegible]

Leonardo Guerreiro Azevedo, D.Sc. – IBM Research - Brasil; UNIRIO

[illegible]

Claudia Cappelli, D.Sc. – UNIRIO

[illegible]

Julio Cesar Sampaio do Prado Leite, PhD. – PUC-Rio

RIO DE JANEIRO, RJ – BRASIL

SETEMBRO 2015

Brandão, Bruna Christina Pinto.
B817 Aspects identification in business process through process mining / Bruna Christina Pinto Brandão, 2015.
91 f. ; 30 cm

Orientadora: Flávia Maria Santoro.
Coorientador: Leonardo Guerreiro Azevedo.
Dissertação (Mestrado em Informática) - Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro, 2015.

1. Mineração de dados (Computação). 2. Gestão de processos de negócios. 3. Linguagem de programação (Computadores). 4. Controle de processos - Administração. I. Santoro, Flávia Maria. II. Azevedo, Leonardo Guerreiro. III. Universidade Federal do Estado do Rio de Janeiro. Centro de Ciências Exatas e Tecnológicas. Curso de Mestrado em Informática. IV. Título.

CDD – 005.74

“If you steal from one author it's plagiarism; if you steal from many it's research.”
Wilson Mizner

“If we knew what it was we were doing, it would not be called research, would it?”
Albert Einstein

Agradecimentos

À minha mãe, que, com muito carinho e apoio, não mediu esforços para que eu chegasse até esta etapa de minha vida.

À minha irmã, que foi fundamental em minha formação pessoal e profissional e que sempre esteve comigo me incentivando e apoiando.

Aos meus orientadores Flávia Santoro e Leonardo Azevedo, pelo convívio, pelo apoio, pela compreensão e pela paciência na orientação e incentivo que tornaram possível a conclusão desta dissertação.

A todo o corpo docente da UNIRIO, pelo carinho, dedicação e qualidade de ensino demonstrado ao longo do curso.

Aos meus amigos, que foram essenciais durante esse período, me auxiliando, apoiando, rindo e me divertindo.

E, finalmente agradeço a todos aqueles que de alguma forma estiveram e estão próximos, me incentivando a sempre a conquistar as minhas loucuras.

BRANDÃO, Bruna Christina Pinto. **Aspects Identification in Business Process Through Process Mining.** UNIRIO, 2015. 91 páginas. Dissertação de Mestrado. Departamento de Informática Aplicada, UNIRIO.

Resumo

Elementos utilizados na modelagem de processos de negócio podem estar dispersos em processos distintos, o que torna difícil gerenciar mudanças, analisar melhorias no processo ou verificar impactos transversais. Estes elementos dispersos são chamados de aspectos. Semelhante ao paradigma orientado a aspectos em linguagens de programação, em BPM, o uso de aspecto tem o objetivo de modularizar os interesses transversais espalhados por todos os modelos. A modularização facilita a gestão do processo (por exemplo, na reutilização, manutenção e compreensão). As abordagens atuais para identificação de aspecto são feitas manualmente, resultando no problema da subjetividade e da falta de sistematização. Este trabalho propõe um método para identificar automaticamente os aspectos em processos de negócios através de seus logs de eventos. O uso de log de eventos foi escolhido por representar o processo como é executado na realidade. O método baseia-se em técnicas de mineração e que tem como objetivo resolver o problema da subjetividade da identificação feita por especialistas. Os resultados do método mostram-se positivos sobre a identificação de aspectos. O método pode ser utilizado para auxiliar especialistas em identificar aspectos.

Palavras-chave: Aspectos, Mineração de Processos, Gestão de Processos de Negócio.

Abstract

Elements used to model business process models can be scattered (repeated) within different processes, making it difficult to handle changes, analyze process for improvements, or check crosscutting impacts. These scattered elements are named as Aspects. Similar to the aspect-oriented paradigm in programming languages, in BPM, aspect handling has the goal to modularize the crosscutting concerns spread across the models. Modularization facilitates the management of the process (*e.g.*, concerning reuse, maintenance and understanding). The current approaches for aspect identification are made manually, resulting in the problem of subjectivity and lack of systematization. This work proposes a method to automatically identify aspects in business process from its event logs. Event logs were chosen because it corresponds to information about how processes are really executed (as-is). The method is based on mining techniques and it aims to solve the problem of the subjectivity identification made by specialists. The results from the method are positive about aspects identification. The method can be used to help specialists identify aspects.

Keywords: Aspects, Process Mining, Business Process Management.

Table of Contents

List of Figures

List of Algorithms

List of Tables

List of Abbreviations

AO-BPMN – Aspect-Oriented Business Process Modeling Notation

AOP – Aspect-Oriented Programming

AOPML – Aspect Oriented Process Modeling Language

BPEL – Business Process Execution Language

BPM – Business Process Management

BPMN – Business Process Model Notation

EPC – Event-driven Process Chain

IDEF – Integrated Definition

NLP – Natural Language Processing

SOA – Service-Oriented Architecture

UML – Unified Modeling Language

WSD – Word Sense Disambiguation

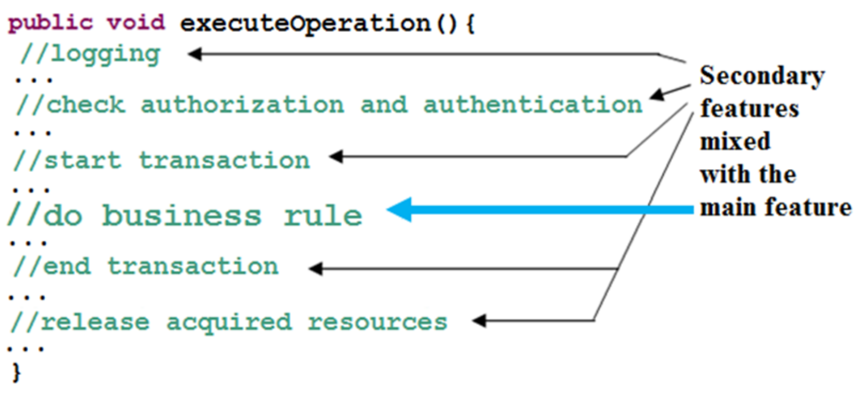
1. Introduction

In this chapter, it is presented the motivation of this work, the problem being addressed, the working hypothesis (*i.e.*, the research question), and the scientific methodology applied. It also presents the dissertation structure.

1.1. Motivation

In a software development, the functionalities are usually implemented through object-oriented programming. In object-oriented programming, each class method aims to execute a given business rule (*i.e.*, primary functionality), but it also contains secondary functionalities which supports the primary functionality in its execution.

The software code corresponding to the secondary functions, such as exception handling, time constraints, transaction management, security control, access control, logging, are many times spread all over an application (*i.e.*, in many methods). For example, a data logging functionality in object-oriented programming is implemented in a single class, but it is invoked in other methods as a secondary function, becoming a spread code. These spread codes are defined as crosscutting concerns [PAHLSSON, 2012]. Figure 1 illustrates secondary functions mixed with the business rule in a code.



```
public void executeOperation(){
 //logging
...
 //check authorization and authentication
...
//start transaction
...
//do business rule
...
//end transaction
...
//release acquired resources
...
}
```

Figure 1- Example of secondary functions with the main functionality [Garcia, 2010].

The aspect-oriented programming (AOP) approach aims to encapsulate spread codes through a construct called Aspect [KICZALES *et al.,* 1997]. An Aspect changes the behavior of the code with an additional behavior at one point in the code execution. This additional behavior is called advice, and the execution point is defined as join point [CASACHI *et al.,* 2012].

The aspect-oriented programming allows the code to be encapsulated and modularized. This programming methodology has advantages, such as reduction in the code spreading, the responsibilities of each module are more transparent, and the modules are more independent. The evolution of the application is easier since the loose coupling enables changes in the application without changing the main functionalities. There is more reusability of code because the modules are less dependent. The systems are easier to develop and maintain because aspect facilitates the integration of new functionalities without causing problems to other parts of the system. Besides, it has a low cost to introduce new features [GARCIA, 2010].

The aspect-oriented paradigm inspired its use in BPM (Business Process Management) with the same goal to modularize the crosscutting concerns spread across the software specification [CAPPELLI *et al.,* 2009]. Crosscutting concerns in BPM are also defined as the interests that cause the scattering and tangling problem. The scattering problem is the repetition of concerns within process models. The tangling problem means that any change in a concern should be reflected in all processes which use the concern [CAPPELLI *et al.,* 2010] [CHARFI *et al.,* 2010] [COLLELL, 2012] [JALALI *et al.,* 2014a] [WANG *et al.,* 2005]. Thus, in BPM, crosscutting concerns are elements scattered and tangled within process models. They can be part of the process core (*e.g.,* business rules), and not only secondary features as in AOP (*e.g.,* exception handling, time constraints, check authorization and authentication, transaction control, security control, access control, logging) [JALALI *et al.,* 2014b] [SANTOS *et al.,* 2011].

Using aspects in BPM improves the modularization of business process model. Thus, the aspect identification (and consequently its implementation) facilitates maintenance, understanding, modeling and the reuse of parts of the process. Hence improving the process management.

1.2. Problem Description and Hypothesis

Aspects identification assists to improve process modularization. It facilitates management of the process (*e.g.*, reuse, maintenance and understanding). A method to identify aspects automatically can assist in decision making and saving time. Currently, the proposals for identifying aspects laid on manually analysis of business process models by experts [CAPPELLI *et al.*, 2009] [JALALI *et al.*, 2014b]. Although based on heuristics, the identification made by humans raises the problem of subjectivity, because it depends on each expert judgment. Towards an answer for this problem – the absence of a more "technical", unbiased identification method – it results in the following research question

`How to identify automatically aspects in business processes?`

The research hypothesis undertaken is:

`If process mining techniques are applied on information systems event logs, then aspects in processes can be identified reducing the subjectivity in the identification.`

In order to address this research question, this work aims at developing an automated method to identify aspects. The method applies process mining techniques on information systems event logs. The identification of aspects in logs makes the technique independent from the process model, although it is dependent on the event log structure. A proof of concept was performed to do a primary evaluation of the method and to demonstrate the potential for real-world application. Afterwards, it was evaluated with a real life case study. The results from the method are positive. In both evaluations – proof of concept and real life case study – the method was capable to identify aspects.

1.3. Scientific Methodology

The scientific methodology employed in this work was conducted in four stages. In the first stage, it was defined the research problem and the hypothesis. Besides, process mining on event logs was chosen as approach for the solution. At the next stage, a theoretical background research was performed, looking for existing techniques or inspiration to be used in the created solution. In the next step, the solution was designed and implemented. In the third stage, a proof of concept was performed for an initial evaluation through a comparison of results from an example of the literature. In this case, it was used the same processes used by Tavares and Marinho [2014] to check if the proposed method identifies the same aspects identified by the experts in [TAVARES *et al*., 2014]. In the fourth stage, it was conducted a case study on a real life log of events. This case study aimed at evaluating the proposed method in a real life context. The results from both studies (proof of concept and real life case study) are presented in the corresponding chapter (Case Study). Figure 2 illustrates the stages and steps.

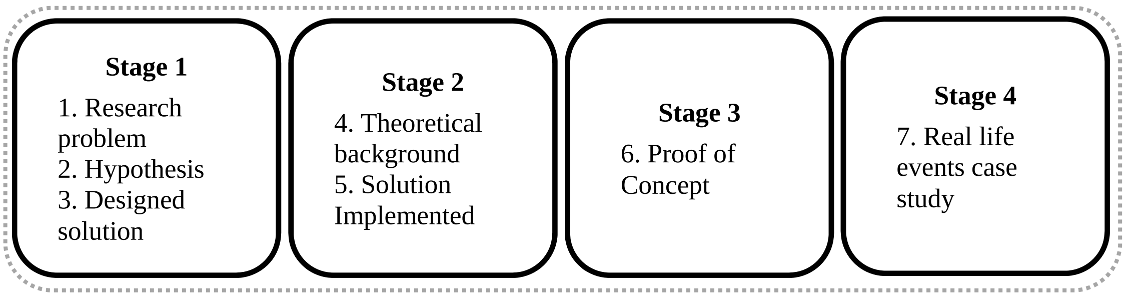


Figure 2 – Scientific methodology overview.

This work is divided in 5 chapters as follows. The Chapter 1 is the introduction. Chapter 2 presents the background concepts, including business process management, aspects, aspects in business process management and process mining. Chapter 3 analyzes the related work. Chapter 4 describes the proposed method. Chapter 5 presents the case study with the proof of concept and the real life case study. Chapter 6 presents the conclusion and list future works.

2. Background

This chapter presents the theoretical background concepts. The concepts of business process management, aspects, aspects in business process management and process mining are explained.

2.1. BPM

According to Davenport (1994, p. 6) “A process is a simply set of structured activities and measures designed to result in a specified product for a particular customer or market. Thus, a process is a specific ordering of work activities across time and space, with a beginning, an end, inputs and outputs clearly identified: a framework for action “[DAVENPORT, 1994].

Weske [2012] defines business process as: “a set of activities that are performed in coordination in an organizational and technical environment. These activities jointly realize a business goal. Each business process is enacted by a single organization, but it may interact with business processes performed by other organizations.” A business process is described by one or more procedures that together perform one business objective. The executing of a business process has a well-defined start and end conditions, and can combine automatic and manual procedures [WfM, 1999].

Business-processes-oriented companies have a horizontal structure, whose main feature is the focus on the customer, operating under a matrix structure, where hierarchical managers are replaced by process owners. They operate with autonomy and responsibility for the entire process, regardless of the hierarchical structure. The horizontal organization, oriented to business processes, makes the operation more flexible, focused on the organization's purposes and with greater proximity to the final consumer [OSTROFF, 1999]. On the other hand, vertical organizations are those structured by functions, in which the hierarchical pyramid is the great feature. Overlapping and superimposed decisions levels are common in this kind of organizations, creating frequent friction among end customers and suppliers.

The business process management is concepts, methods and techniques in which supports the design, administration, configuration, enactment and analysis of business processes. The base of business process management is the explicit representation of processes with their activities and the constraints between them. Business process management supports analysis, improvement and enactment of the processes [WESKE, 2012].

The business process modeling is the set of practices or tasks that companies can execute to visually describe all aspects of a business process. The aspects includes its course, control and decision points, triggers and conditions for the executing of activities, context in which an activity runs and the associated resources [JOSUTTIS, 2007 apud BLOOMERG *et al.*, 2006].

A model is a representation of the process that enables companies to document, simulate, share, implement, evaluate and continuously improve their operations [JOSUTTIS, 2007 apud BLOOMERG *et al.*, 2006].

The activities of analysis and process modeling can be performed using tools available on the market. There are about 300 software that offers a variety of features depending on the chosen product [OLIVEIRA *et al.*, 2006]. Examples of tools are: Aris (http://www.softwareag.com/corporate/default.asp), BizAgi (http://www.bizagi.com), Bonita (http://www.bonitasoft.com) and Signavio (http://signavio.com). A business process management system is a generic software system that is driven by explicit process representations to coordinate the enactment of business processes [WESKE, 2012].

Among the currently most widespread notations for process modeling are BPMN (Business Process Modeling Notation), UML (Unified Modeling Language), IDEF (Integrated Definition) and EPC (Event-driven Process Chain). After choosing the notation and tool, you must identify the process or processes you want to model conducting a detailed survey of the process to discover the workflow, who is responsible for initiating the process, who makes the next activity, among other details [VALLE *et al.*, 2012].

2.2. Aspects

Originally, aspects are the modularization of crosscutting concerns in object-oriented software. The aspect-oriented programming allows the code to be encapsulated and modularized.

The implementation of aspect-oriented programming normally consists of [SOARES *et al.*, 2012]:

- A component language for components programming;
- One or more aspect languages for aspects programming;
- An aspects combiner (aspect weaver) – to combine the artifacts of the programming language and the aspects;
- A program written in the components language;
- One or more programs written in the aspects language

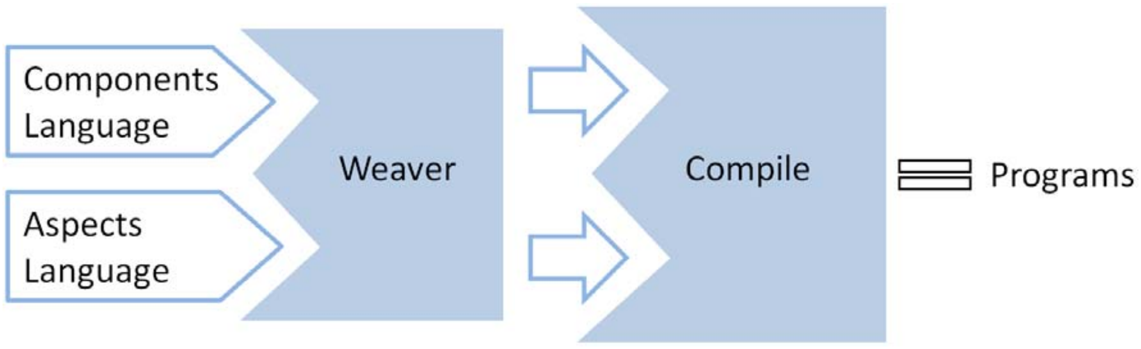


Figure 3 – Weaver relationship [adapted from [SOARES *et al.*, 2012].

Figure 3 illustrates the relationship of the artifacts present in the aspect implementation.

In the context of aspect-oriented programming, components are abstractions provided by the language that allow the implementation of a system functionality (procedures, classes, functions, objects) [SOARES *et al.*, 2012].

The aspect language supports the implementation of desired features clearly and succinctly, providing the programmer structures needed to describe the behavior of aspects and situations in which they must act [SOARES *et al.*, 2012]. AspectJ is an example of an aspect language. It extends the Java language with new structures to support the modular implementation of crosscutting concerns. New elements are:

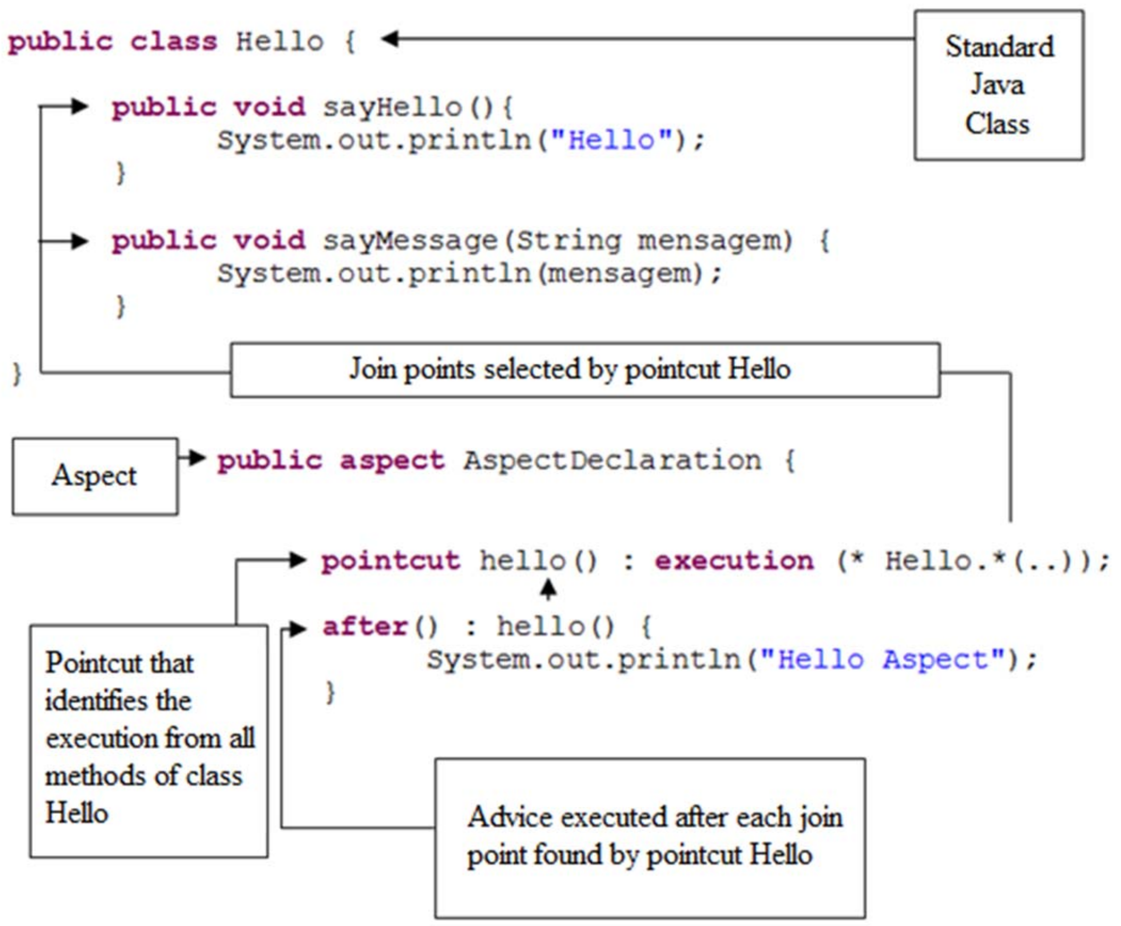
(i) join points represent execution points;

(ii) pointcuts represent sets of join points;

(iii) advice represents the methods attached to pointcuts.

Aspects are modular units of crosscutting implementations composed of pointcuts, advice and own statements of the Java language [KICZALES et al., 2003].

Figure 4 illustrates a Java class using aspect elements. The Java class Hello has two join points sayHello and sayMessage. They will be intercepted by the aspect AspectDeclaration, which has the pointcut hello referencing all methods of Hello class. This pointcut has an advice, which defines that the command hello should be to be executed after the accomplishment of the join point found by the pointcut [GARCIA, 2010]. In other words, after the execution of each method of Hello class the hello method should be executed. The text printed in this system console would be: “HelloHelloAspect”. The combination of the Java code with the Aspect code is done in a process named as weaving.



```
public class Hello {
    public void sayHello(){
          System.out.println("Hello");
    }

    public void sayMessage(String mensagem) {
          System.out.println(mensagem);
    }
}

public aspect AspectDeclaration {

       pointcut hello() : execution (* Hello.*(..));

       after() : hello() {
            System.out.println("Hello Aspect");
       }
```

Standard Java Class
Join points selected by pointcut Hello
Aspect
Pointcut that identifies the execution from all methods of class Hello
Advice executed after each join point found by pointcut Hello

Figure 4 – Relationship between a class and one aspect [adapted from [GARCIA, 2010].

Aspects have similar meaning as services in a SOA (Service-Oriented Architecture) context. Services are defined as parts or entire functions of a system that may be available to another system and can be invoked through messages [JOSUTTIS, 2007]. The services must operate in an independent way of other services, and should have a well-defined interface. Services are logical representations of elements (e.g., activities, business rules, business requirements) in the business process that can be mapped on input, processing and output. When those elements can be performed automatically or by human using systems (in a semi-automated approach), they can be made available as services in SOA. Then the difference from service to aspect is that a service represents elements of business process that can be performed automatically or supported by systems. On the other hand, aspects can also be elements performed manually.

2.3. Aspects in BPM

The same logic of aspects in software code applies to aspect in process models. Crosscutting concerns in BPM are also defined as the interests that cause the scattering and tangling problem [CAPPELLI *et al.,* 2010] [CHARFI *et al.,* 2010] [COLLELL, 2012] [JALALI *et al.,* 2014a] [WANG *et al.,* 2005]. They are elements scattered and tangled within process models, and they can be the part of the core of the process (e.g., business rules) and not only secondary features as in aspects in oriented-programming.

Cappelli *et al.* [2009] propose a meta-language called Aspect Oriented Process Modeling Language (AOPML), which is independent of any business process language. The use of this meta-language improves business process model modularity. In another work, Cappelli *et al.* [2010] developed a notation of AOPML specifically to handle BPMN (Business Process Model Notation – [OMG, 2011]) models. They called this particular notation as Aspect-Oriented Business Process Modeling Notation (AO-BPMN), which uses aspect-oriented paradigm to extend the BPMN. In summary, they proposed the introduction of graphical elements in the notation through new roles for the lanes and relationships. In that case, the crosscutting concerns are represented separated in a vertical lane, orthogonal to the processes. It uses a new connector called relationship cross, which represent the composition between the transverse elements of interest and process model. However, this approach has proved to be confusing for analysts, due to the many lines connecting elements, so there are some works which are trying to improve this representation.

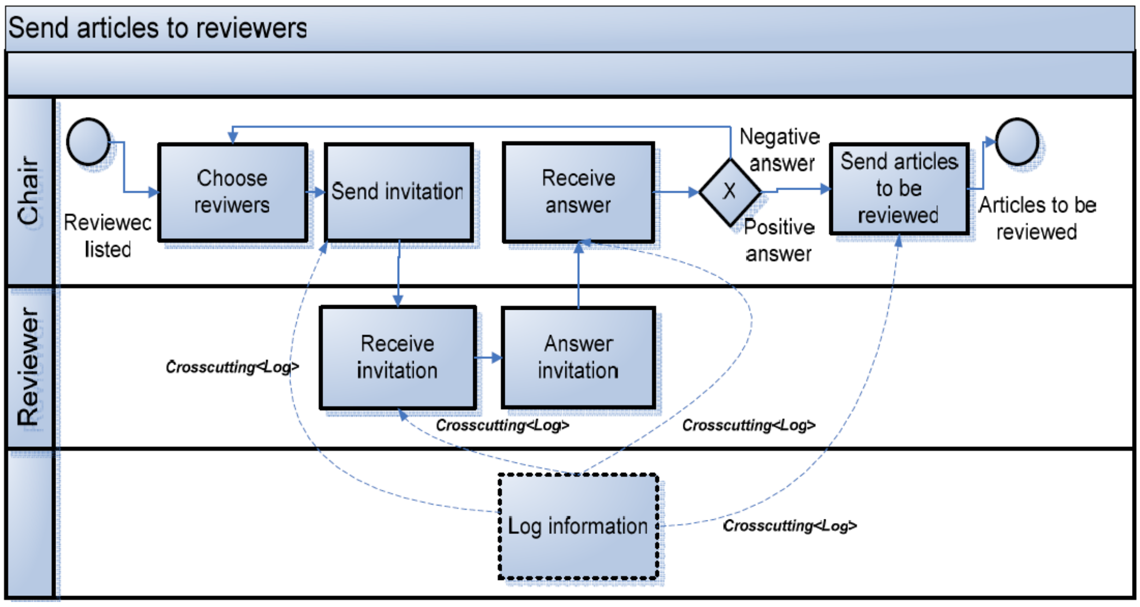


Figure 5 – Process Model with aspects modularization [CAPPELLI *et al.*, 2009].

Figure 5 illustrates the business process model "send articles to reviewers" modularized by aspects using AO-BPMN. In this process, the aspect "Log information" is reused when there is the need to store information, e.g., in the execution of the activities "Send invitation", "Receive invitation", "Reply invitation" and "Receive response". In each activity execution, the aspect "Log information" activates its own execution intervening in the activity within the process, and storing information about the execution of the activity.

The modularization of processes through aspects brings the same benefits of aspect-oriented programming. Processes modularization provide a better understanding of the process because it groups activities by their goals and enables reusability of parts between processes that share the same "interests", including key features such as business rules.

The identification of aspects is the first step to be done before the implementation or the modeling of aspects. Cappelli *et al.* [2010] propose a set of heuristics (first recommended by Silva [2006] in her PhD thesis to support the identification of a concept as a crosscutting concern. The heuristics are:

- "If the concept is repeated several times in different places;
- If the concept is used by different other concepts;
- If the concept reflects an integration of semantically distinct situations;

- If the concept represents a decision situation from which different options may be taken, and its absence does not interfere with the global objectives of the whole;
- If the concept can be reused in other domains; and
- If the concept is very much independent of the other concepts."

The proposed method uses these heuristics as the definition of crosscutting concerns. Therefore, the method seeks events in the log that are repeated several times in different processes. It seeks events which are used by other events and events that can be reused in other processes. It also seeks for an integration (grouping) of crosscutting concerns from distinct situations. That is, if two crosscutting concerns (events) are almost used together from different situations (processes), the method suggests integration. The heuristics "if a concept is independent of other concepts" and "if the concept represents a decision situation with different options, which does not interfere with the global objective" were not explored in the method proposed in this dissertation.

In software development, there are three approaches in to identify aspects: AOP clone, cluster and analysis fan-in. However, there is not a technical approach to identify automatically the crosscutting concerns in business process. Most approaches are manual – executed by experts. The next chapter presents the existing approaches of aspects in business processes.

2.4. Process Mining

Process mining is a discipline that relates data mining and process intelligence. Process intelligence is the discovery, analysis and verification if the process is effective for business improvement. Process mining is performed from data recorded in event logs, containing information that was created during the execution of the process [VAN DER AALST *et al.,* 2004]. Therefore, a process discovered through mining event logs is called AS-IS process, *i.e.,* how the process really runs. On the other hand, a modeled processes is called TO-BE process, *i.e.,* it presents how the process should run [DUMAS *et al.,* 2013].

There are three approaches of process mining: process discovery, process analysis and process verification. The most common algorithm for process discovery is the alpha algorithm [DUMAS *et al.,* 2013]. Alpha algorithm discovers the process by mapping the relationships between the activities present in the process according to their orders in the event logs. Initially, it maps the basic relations of precedence, then the causal relationships, the potential parallelism relations and relations with no direct succession.

The second approach which uses process mining is process performance analysis. The analysis is based on four dimensions: Time, Cost, Quality and Flexibility. The time dimension checks the lifecycle of activities and the waiting time to execute them. The cost dimension can be analyzed if there are expense details present in the logs, *e.g.*, the financial cost or resource time required to execute the process activities of the current instance. The quality dimension analyzes the process quality. The flexibility dimension is the range of variation the process enables, *i.e.*, it analyzes whether the discovered process (as-is) allows paths which are or are not desirable [DUMAS *et al.*, 2013].

The third approach is the process conformance verification, in which the process is assessed regarding its constraints of exclusivity, order and mandatory issues. The constraint of exclusivity checks activities that exclude others activities – *e.g.*, “order accepted” and “order rejected”. The constraint of order checks the order of the activities (*e.g.*, rent equipment before checking availability). The mandatory constraint checks activities which must be executed for any purpose of the organization, *e.g.*, “Review request for rent in order to control the cost”.

Process mining is applied in events log generated by information systems. An IEEE process mining task force created a standard log format to be able to be use in any process mining tool. Van der Aalst *et al.* [2011] standardized XES (www.xesstandard.org), a standard logging format that is extensible and supported by the OpenXES library and by tools such as ProM, XESame, and Nitro. The metamodel of XES is illustrated in Figure 6.

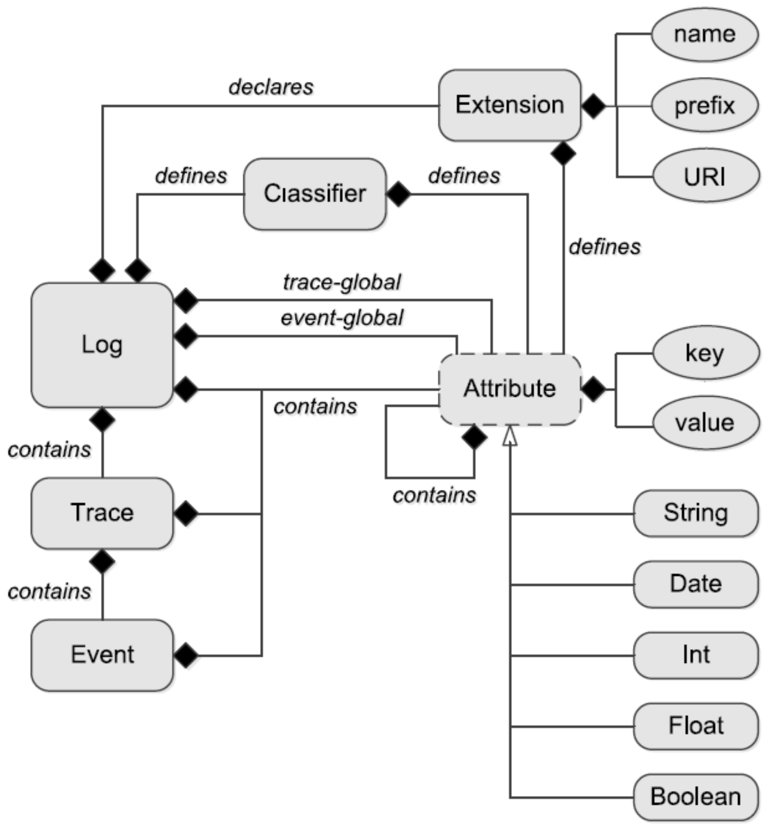
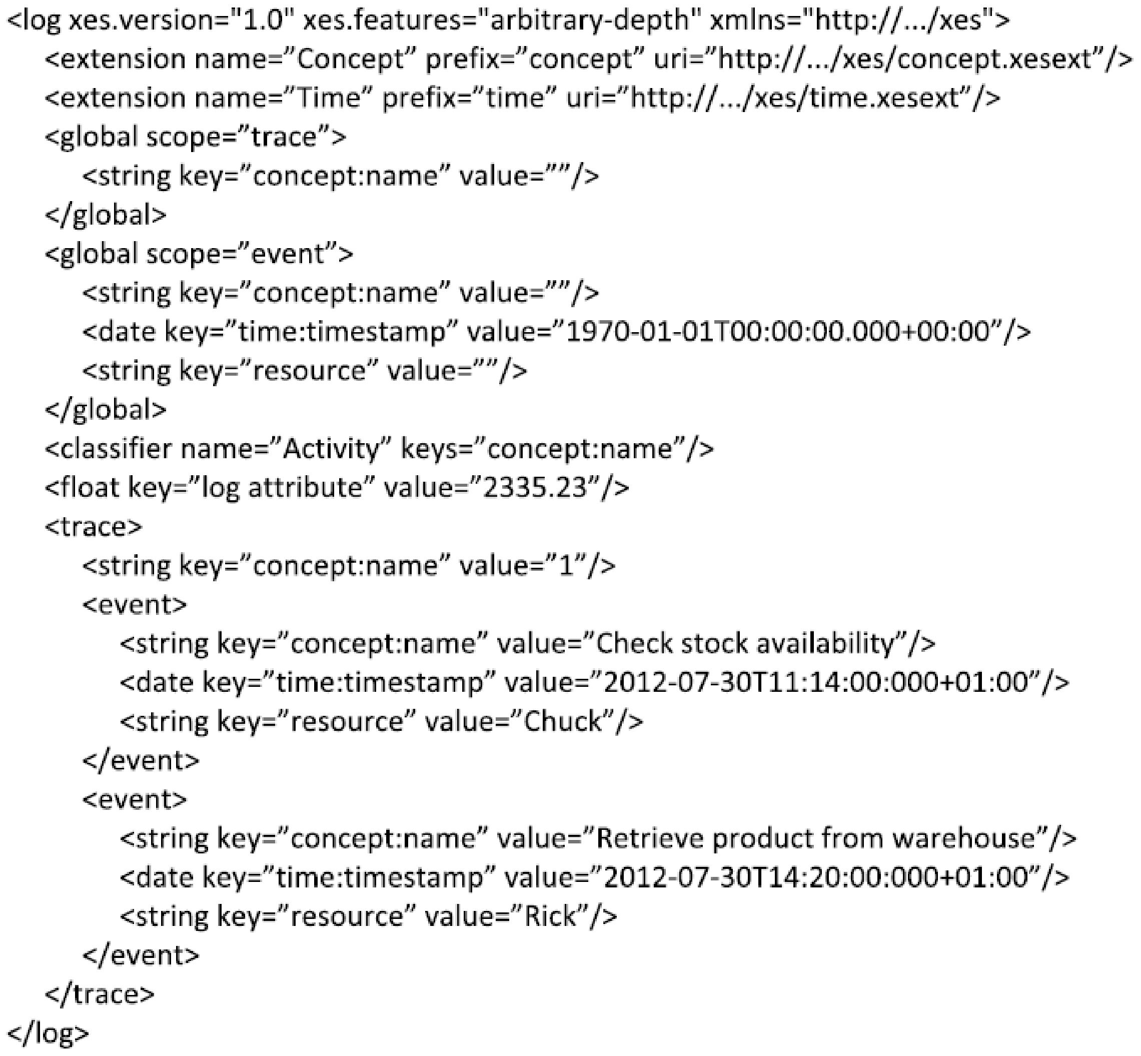


Figure 6 – XES Metamodel [DUMAS *et al.*, 2013].

Each XES file represents one log. Each log contains multiples traces, which contains multiple events. Log, traces and events contain attributes. An attribute corresponds to a key value pair. The attribute value has type, which can be String, Date, Int, Float or Boolean. Attributes refers to a global definition. For example, a global definition could state each trace must have the attribute name. Another global definition could state each event must have the attributes name, timestamp and resource. The classifier maps one or more attributes of an event to a label that is used in the output of the analysis tool [DUMAS *et al.*, 2013]. For example, the log has a classifier defining that the attribute name is an activity classifier, which means that each event in the log with a name will be classified as an activity in the analysis tool. Figure 7 shows a XES file example.



```
<log xes.version="1.0" xes.features="arbitrary-depth" xmlns="http://.../xes">
    <extension name=”Concept” prefix=”concept” uri=”http://.../xes/concept.xesext”/>
    <extension name=”Time” prefix=”time” uri=”http://.../xes/time.xesext”/>
    <global scope=”trace”>
        <string key=”concept:name” value=””/>
    </global>
    <global scope=”event”>
        <string key=”concept:name” value=””/>
        <date key=”time:timestamp” value=”1970-01-01T00:00:00.000+00:00”/>
        <string key=”resource” value=””/>
    </global>
    <classifier name=”Activity” keys=”concept:name”/>
    <float key=”log attribute” value=”2335.23”/>
    <trace>
        <string key=”concept:name” value=”1”/>
        <event>
            <string key=”concept:name” value=”Check stock availability”/>
            <date key=”time:timestamp” value=”2012-07-30T11:14:00:000+01:00”/>
            <string key=”resource” value=”Chuck”/>
        </event>
        <event>
            <string key=”concept:name” value=”Retrieve product from warehouse”/>
            <date key=”time:timestamp” value=”2012-07-30T14:20:00:000+01:00”/>
            <string key=”resource” value=”Rick”/>
        </event>
    </trace>
</log>
```

Figure 7 – XES file example [DUMAS *et al.*, 2013].

The minimum information required in the log for process mining is a trace id, timestamp and activity (or log action). This work assumes the log has at least this information and it fits the XES structure. The XES meta-model has been purposefully and carefully designed to be independent of any implementation. OpenXES is an open source (free software) library implemented in Java, which has been designed to adhere to the following goals [GÜNTHER *et al.*, 2014]:

- To be compliant to the XES;
- To be straightforward to use and easy to integrate by developers;
- To provide the highest performance for event log data management and storage;
- To serve as clear and understandable reference implementation for other implementations of XES.

There are several software products with process mining capabilities, for example: ProM, Disco, ARIS Process Performance Manager and others [VAN DER AALST *et al.,* 2011]. ProM is the only open-source framework and it also has countless process mining algorithms [VERBEEK, 2010]. Disco is a professional complete process mining toolkit from Fluxicon that makes process mining fast and easy, and it has academic license [GÜNTHER *et al.,* 2012]. ARIS Process Performance Manager is a product from Software AG, which has the goal to execute optimization combining business intelligence (BI) with automatically process discovery.

The method proposed in this dissertation is based on two algorithms of process mining. The first algorithm was created by Buijs *et al*. [2013], which identifies a single process model from different event logs. This algorithm uses the open XES library and it has at least two log files as input.

The second algorithm is the Pattern Abstraction algorithm from the plugin with the same name [VERBEEK, 2010]. The Pattern Abstraction is a ProM tool plugin, which automatically discover patterns of activities and suggests grouping the identified activities as a single one, making the process more abstract. This algorithm has as input one log file. The algorithm created in this dissertation to find patterns used the idea to group events by its pattern, but with a different implementation. The implementation is different because the method has at least two logs as input and it does not use the abstraction functionality. The details from the algorithms used in this dissertation is described in Chapter 4.

3. Related Work

The proposals for identifying aspects in business process are in general based on manual identification by experts [CAPPELLI *et al.,* 2009] [JALALI *et al.,* 2014b]. Other approaches for aspect identification analyzed are on the domain of software requirements, not in business process models. They identify aspects automatically in text, but they only consider non-functional aspect of business process instead of functional aspects as the approach of this work. Those approaches for automatically identification of aspect in text consider keywords and document structure to identify crosscutting concerns, *i.e.,* it searches for verbs referring to verifications ("checks", "verifies") or verbs associated with persistency ("adds", "stores", "updates" and "modifies") in specifics places in the use case (*e.g.* descriptions) . Hence, they do not consider functional crosscutting concerns, *e.g.,* business rules.

Jalali [2014b] proposes an approach to enable the identification of process models with crosscutting concerns from the event logs. He states the process mining techniques should discover crosscutting concerns from a log file by searching for duplicates tasks. However, the subject of finding duplicate task within a process is in an early stage. For example, the existing techniques cannot claim if the task is duplicated or if the process has loops. He also states that there is no existing solution to discovery crosscutting concerns from a log file of multiples process models. This is due to the lack of process mining techniques able to relate and analyze process variants. Thus, the discovery of multiple processes models with crosscutting concern within one log file is not possible yet [JALALI, 2014b apud VAN DER AALST, 2013]. After demonstrating the current techniques of process mining cannot discover processes models with crosscutting concerns, he proposes an approach to discover this kind of processes models.

The Jalali [2014b] proposed approach – to enable discovering process models which contain crosscutting concerns – is a cyclical process containing four phases: (i) Identification of crosscutting concerns; (ii) Elimination of crosscutting concerns; (iii) Business process discovery; (iv) Crosscutting concern and process model relation discovery. At the first phase, the identification process is manual by doing interviews with specialists. At the second phase, the crosscutting concerns identified by the first phase are eliminated from the event logs. Therefore, the logs will only contain the main activities for each process model. At the third phase, the discovery of business process model is done by process mining algorithms over the cleaned log. At last, in the fourth phase, the relation between the crosscutting concerns and process model are found. This relation discovery is also done by interviewing the specialists. The whole approach is cyclical, so all crosscutting concerns may be identified and their events may be eliminated through several cycles. Even though he mentions automatically discovery by mining techniques to discover duplicated activities, he makes the aspect discovery manually by interviewing business process specialists, and not automatically as the proposal of this work.

Sampaio *et al.* [2005] propose the identification of aspects using natural language processing (NLP) in business requirements. To the best of our knowledge, these techniques are reported as ineffective when requirements are complex or unstructured. Their work presented an approach for mining aspects from requirements-related documents based on NLP techniques that enable an efficient context sensitive analysis of textual documents. The documents can vary from very informal textual documents, such as interviews and high level descriptions of the system, to more structured documents such as use case textual descriptions or viewpoint descriptions. A tool was developed to provide support and help the developers automatically mine and model the crosscutting concerns without having to previously read the requirements documents.

Campos *et al.* [2010] discussed the identification of aspects via inspection of use cases descriptions. Their technique seeks identify aspect candidates analyzing a set of use cases described in a specific format (template). They suggested a use case template and a checklist to verify consistency of the process description. The template includes sections to make explicit aspect candidates, *i.e.*, use case step numbering indicates where the aspect candidates should be inserted (*i.e.*, join points). Their work emphasizes the aspects candidates are normally found in the extensions of the description. The main difference between the approach and others is the use of the description structure to identify aspect candidates. Also, it is the only work about mining early aspects in business requirements that considered both requirements (functional and non-functional).

Rago *et al.* [2009] propose also the identification of aspects using natural language processing techniques from use cases descriptions. They present a semi-automated approach, which aims at improving the precision of the aspect identification process in use cases. They apply a combination of text analysis techniques (natural language processing – NLP) and word sense disambiguation (WSD). Their analysis is focused on the relationships among terms in use cases (*e.g.*, verbs, direct objects) that often hint crosscutting behaviors. The approach generates a graph of candidate concerns that are scattered in the use cases. It also produces a ranking of these concerns according to their importance. Then, the developer selects which concerns are relevant. Although the approach can be integrated into a UML development improving requirements elicitation, the processing time is a challenge. The better the approach precision, more time would it take (precision of 90% took 90 minutes). Their approach suggests the techniques (NLP and WSD) can solve issues related to synonyms, vagueness and ambiguity in text.

Souza *et al.* [2011] recommended the identification of services, considering a SOA (Service-Oriented Architecture) approach, from a business process model designed in an aspect-oriented fashion (AO-BPM) [CAPPELLI *et al.,* 2010]. Services should be modular [JOSUTTIS, 2007]. This allures modularization as a key concept in service identification. Thus in an aspect-oriented business process modeling (AO-BPM), crosscutting concerns are encapsulated in modules (aspects) improving the modularity of the process. Therefore, service identification will benefit from modular business processes. The approach is based on an existing method for service identification without aspect modularization [AZEVEDO *et al.*, 2009 and 2011]. They describe the evolution of the existing method to achieve the goal of using the aspect-oriented business process for identifying candidate services. There are other approaches for service identification from business process models and for service composition from an aspect-oriented approach [CHARFI *et al.,* 2004]. However, all of them are concerned with services identification and not with aspect identification.

Tavares *et al.* [2014] propose improvements to the business process modeling notation originally published by Cappelli *et al.* [2010], which uses the aspect orientation, generating smaller models for easy understanding and maintenance. Furthermore, they propose a guide for aspect-oriented process modeling. Their work is based on the three main elements of aspects (joinpoint, pointcut and advice). The joinpoint is represented by a small circle with the number of the aspect in the model. The advice is a textual representation to help the execution of the artifact attach to the joinpoint. The pointcut is also a textual representation which shows where the aspects are executed. It is represented by the pool side, which illustrates the activities related to aspects. The aspect modelling is done in a separated pool from the business process model. This change has been proposed because the AO-BPM works well for small models, but for large models, it can hinder reading and understanding. The aspect pool should follow some rules. The first rule is that it must be created one lane for each aspect, which in turn must have the artifact that identifies which aspect will be used. The second rule is that the actors should be identified in divided lanes. The third rule is that the aspect pool should be independent from the process model, and the output present in the model. The guide for aspect-oriented process modeling extends a process modeling methodology found in the literature ([SHARP *et al.,* 2001]). The methodology goal is to gather information from the process model to be able to outline and understand it. The outline of the process is to gather all possible information to define the process limits (where begins and end), main activities, actors, stakeholders, data and system involved. It consists by 6 steps:

- Assess the company culture – Indicate the reason of the existence of the business, what the company does and what it is.
- Develop a process map – Identify a set of related processes and draw up a map in order to clarify what is inside/outside of the scope of each one of them, and it also shows the relations between processes.
- Set the scope of the processes – Establish the scope of each one of the process and does a record with the most important information.
- Assess the stakeholders – Realize stakeholders that relate to aspects and not a main activity.
- Investigate the enablers of the process – For example: Process workflow design (actors, steps and flow); Motivations and measurements; Policies and Rules; and others.
- Develop a glossary of terms;

This results in documentations, which is used to identify the aspects during the outline step. The aspect identification is done manually throughout the creation of the documentation. However the aspect identification from the method in this dissertation is automatically from event logs.

Although there are approaches for aspects identification in business requirements and use cases ([CAMPOS *et al.*, 2010], [RAGO *et al.,* 2009], [SAMPAIO *et al.*, 2005]), and to use aspects to identify services [SOUZA *et al.*, 2011], the approach from this work differentiates by identifying aspects from events logs. The main benefit of this approach is to identify aspects automatically in the business level, *i.e.*, in the business process models, the obvious crosscutting concerns (non-functional requirements) normally are not present. The method tries to identify the scattering and tangling problem within a process execution focusing on the functional requirements instead of only searching for non-functional requirements. Besides this advantage, the method proposed in this work is not totally dependent on the specialist judgment. In the analyzed works, even though the specialists follow heuristics, the manual identification leads to the subjectivity problem.

Jalali [2014b] work is the only one that resembles to the method of this work. Although Jalali is identifying process models through process mining, then after the model discovery, he identified aspects in the model. The goal of the method of this work is to identify crosscutting concerns and not the process. Table 1 presents a comparison of these works.

Table 1 – Related work comparison.

Work	**Manual identification?**	**Automatic identification?**	**Artifact of identification?**	**Identification among processes?**
Jalali [2014b]	Yes (by specialist).	No (discusses manners automatically, but does manual identification by specialist).	Uses process mining to discover the process from event logs. However, aspects are identified using the models instead of the logs.	No. Identification within one process.
Sampaio *et al.* [2005]	No	Yes (using NLP)	Requirements documents (complex or unstructured)	No. Different documents, but the same system.
Campos *et al.* [2010]	Yes (inspection of use cases descriptions)	No.	Use cases descriptions.	No. Different use cases, but aspects only found in a single use case.
Rago *et al.* [2009]	No	Yes (using NLP and	Use cases descriptions.	Yes. Looks for concerns

		WSD)		scattered among use cases.
Souza *et al.* [2011]	Yes (A specialist identifies services, not aspects)	No	Process model designed using AO-BPM.	No. Services identified within a single process.
Tavares *et al.* [2014]	Yes (by specialist).	No	Documents with process information	Yes. Seeks for activities repeated in different processes.
Method from this work	No	Yes	Process log (event logs)	Yes, however is restrict to identify aspects among processes.

4. Aspect Identification Method

In this chapter, it is presented the aspect identification proposed in this dissertation. All the steps and algorithm used in the method are detailed.

4.1. Method Description

In the literature, the identification of aspects in an aspect-oriented programming is most achieved by three approaches (Clone, Cluster and Fan-in analysis) [Barbosa, 2008]. The Clone approach aims to discover the methods in the code that can be clones of others, *i.e.*, duplicated code. The Cluster approach aims to group patterns of methods by their execution. For example, if the methods A, B and C are always performed together, they can be grouped as an aspect candidate. Finally, the Fan-in approach calculates the fan-in measure of methods to identify aspects, *i.e.*, the approach counts the number of times the method is invoked by other methods, and when a method is invoked more times than a defined threshold, it is identified as an aspect candidate.

The approach of this work is to identify aspects in business process model inspired in the first two software engineering methods (clones and clusters). So, the proposed method is to identify aspects automatically from event logs by the clone and cluster approach. The Fan-in approach was not used as inspiration in process model because it is difficult to define if the frequency of an event in the event log is meaning that the process has a loop or if it really has a significant meaning as a candidate aspect.

The proposal is based on three assumptions.

- **Event logs are structured in XES format**. This choice was made because XES is the standard log format defined by the IEEE process mining task force. The standard log format was created to be able to be used by any process mining tool [Van der Aalst *et al.*, 2011].
- **The number of different business process logs should be equal or higher than two**, *i.e.*, the method only discovers aspects spread among processes, not within the same process. So, the assumption is that there is a minimum of two logs. This assumption was made because the method looks for aspects among different process logs. In order to address

aspects within the same process (i.e., the same process log), it would be necessary a wide study using natural language processing. It would be necessary to understand the context of the process and the meaning of the activities to try to classify process elements as crosscutting concern – to be able to identify aspects by semantics. However, it would be still hard to understand whether an event is crosscutting the process or the event is inside a process loop.

- **The logs must present the same level of abstraction of the elements**. For example, if a process about acquiring products has the activities “go to supermarket”, “buy items in the list”, and “return home”, and the another related process has the activities “park car in supermarket lot”, “get cart”, “put milk in cart”, “put bread in cart”, “go to cashier”, and so on, then the proposed method will not be able to identify aspects due to the level of the process abstraction (process details) is different. This assumption exists because the method does not calculate the similarity of words to discover if one word might have a more abstraction meaning than another. For example, in Figure 8, the word “items” from the second activity can be an abstraction for the words “milk”, ”bread” from Figure 9. Besides, similarity calculation between words, to be able to consider abstraction levels, the method would have to discover how an action from one activity can mean a set of activities from other processes. For example, the action “buy” from the activity “buy items in the list” from Figure 8 can be in an higher abstraction level of activities “get”, ”put”, “go” from the activities “get cart”, “put milk in cart”, “put bread in cart”, “go to cashier” from Figure 9. Thus, to simplify the implementation of the method, the functionally of similarity calculation was not added in the method first version. Therefore, the method does not control the input; the success of the output is dependable on the user input.

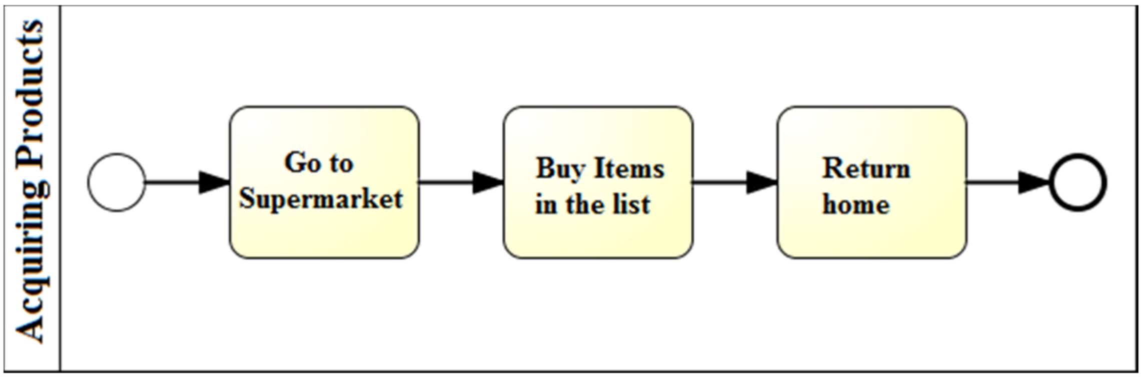


Figure 8 – Acquiring products process more abstract.

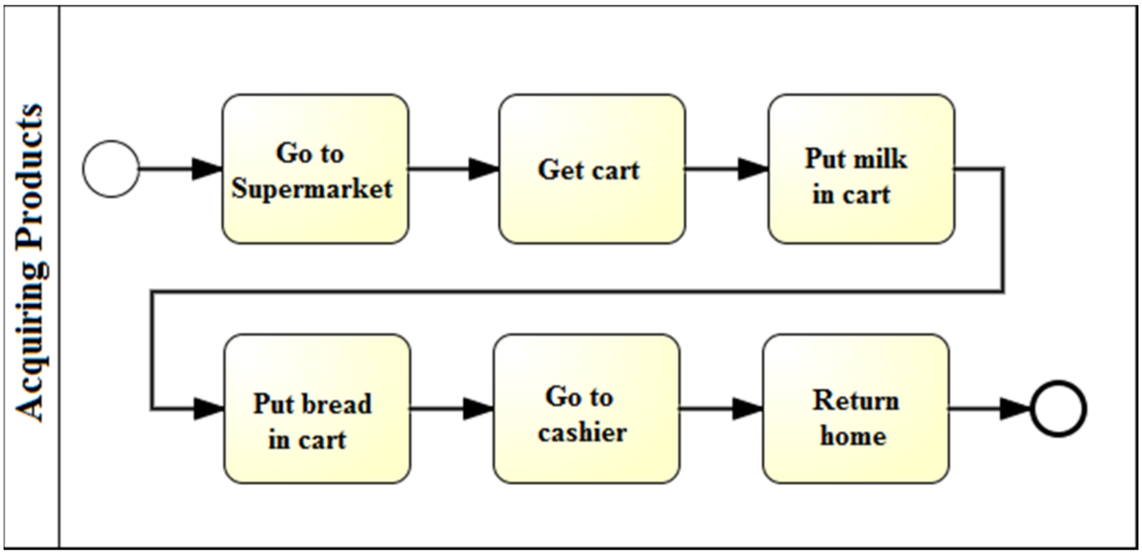


Figure 9 – Acquiring products process less abstract.

Figure 8 and Figure 9 illustrate the same process; however, they have different abstraction levels. For example, the activity “Buy items in the list” from Figure 8 contain the activities “Get cart”, ”Put milk in cart”, ”Put bread in cart” and ”Go to cashier” from the process of Figure 9. Neither one of the processes is wrong; they only have their descriptions in different details. Therefore, the method proposed in its first version has the limitation of not being able to identify aspects from different level of abstraction.

4.2. Method Overview

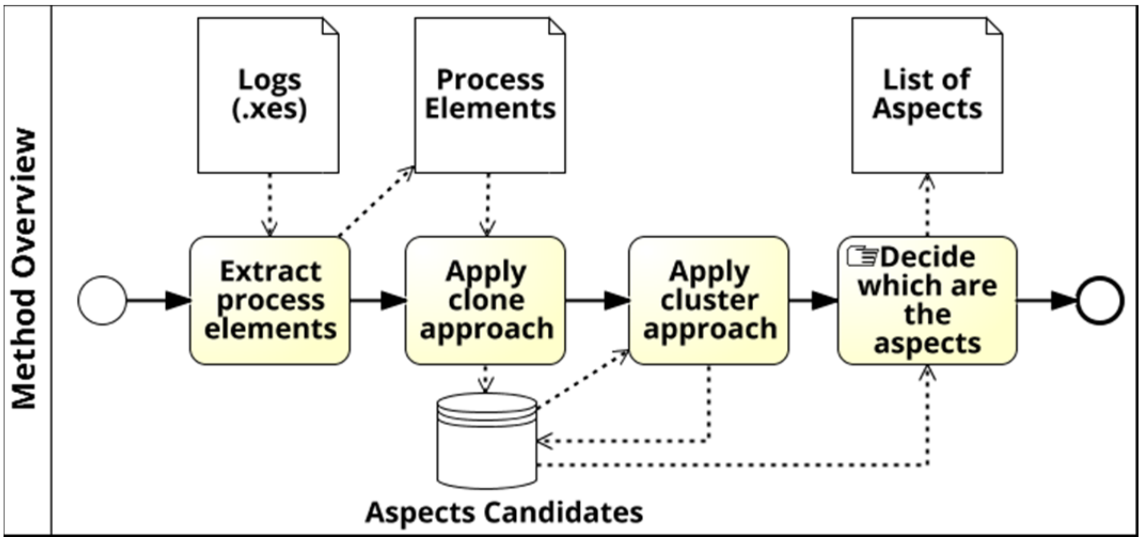


Figure 10 – Method Overview.

Figure 10 depicts an overview of the method, which first reads the logs and extracts the process elements. The logs are in the XES format, which it is the standard log format created by Van der Aalst *et al.* [2011]. Afterwards, it applies the clone approach to identify aspect candidates considering elements with same names or synonym names. Synonym names are analyzed by using natural language processing and WordNet. The comparison between processes is conducted by pairs. Afterwards, the third activity is to apply the cluster approach, which identifies aspect groups considering the order of execution of the aspect candidates identified by previous step. For example, two or more aspect candidates that are executed in sequence are identified as an aspect group if the group appears more than a threshold percentage defined by the user. The final activity is a manual action done by a specialist to decide which aspects candidates is really an aspect considering the list of aspects candidates identified by the method. Process mining has reached a certain level of maturity and it has been used in a variety of real-life case studies, but it still lacks a common framework to evaluate process mining results. A common means of assessing the results was not developed. There is no framework to enable a comparison of performance from the process mining algorithms and to end users evaluate the validity of their process mining results [ROZINAT *et al.*, 2008]. Therefore, any process mining techniques needs to be endorsement by specialist to have some validation.

4.2.1. Extract Process Elements

The method begins reading the logs using Open XES, an open source library in Java to store and manage event log data [GÜNTHER *et al.*, 2014]. The implementation of the use of this library in Java was based in the implementation of the algorithm presented by Buijs *et al*. [2013], which identifies a single process model from different event logs.

The output of this step is a list of unique process elements by log. The algorithm runs over the XES structure (Section 2.4). The goal is to get the name of each event from each trace from each log. Therefore, the algorithm runs over a list of logs in the XES structure extracting the process elements of each log. In the end, the algorithm returns a list where each position is a list of events of a log. The algorithm is described in Algorithm 1. In Algorithm 1 the variables with a capital letter X is from the XES file, that is, it is from the open XES library.

Algorithm 1 – Read XES Files and extract process event names.

```
Input: List of path to logs files (listPath)
Output: Set of logs with its unique events names (listLogs)
1   Initialize a variable from XES library (xes)
2   Foreach path p in listPath do
3   |  Get system current time
4   |  Set of XLog structure listXLog = Call xes.readLog(p) function
5   |  Create list of log model objects listLog
6   |  Foreach xLog in listXLog do
7   |  |  Create list of events names (listEvents)
8   |  |  Create an Iterator XTrace traceIter = xLog.iterator()
9   |  |  While traceIter has next
10  |  |  |  Create a XTrace variable xTrace = traceIter.next()
11  |  |  |  Create an Iterator of XEvent eventIter = xTrace.iterator()
12  |  |  |  While eventIter has next
13  |  |  |  |  Create a XEvent variable xEvent = eventIter.next()
14  |  |  |  |  Events name = xEvent.getAttributes().get("concept:name")
15  |  |  |  |  If listEvents doesn't contains name
16  |  |  |  |  |  Add name to listEvents
```

```
17  | |  Add *listEvents* to *listLog*
18  **Return** *listLog*
19  Get system current time
```

Algorithm 1 – Read XES Files and extract process event names.

4.2.2. Apply Clone Approach

After loading the logs, the method finds the events that exist in at least two logs by its names. Since it is probable that equal events has synonym names, it was implemented the clone approach (discovering semantically similar methods) with natural language techniques. This method was inspired by the technique proposed by Richetti *et al.* [2014], which uses natural language processing to discover process activities that have similar meaning in order to simplify the declarative process models. This approach does not take in consideration elements that are semantically similar, but only synonyms because of the abstraction problem. For example, if there is “Send invitation” and “Receive invitation” activities in the business processes, the method verifies if the words “send” and “receive” are synonyms. The method would not classify these activities as aspects because the words are not synonyms even though the activities’ names are semantically similar.

The first algorithm of this phase (Algorithm 2) has the goal to tag the events names by natural language processing (NLP) tags, that is, tagging words to its grammatical classes (verbs, nouns, adverbs, adjectives, prepositions and others). This algorithm runs over the list of logs resulting from Algorithm 1 getting unique events names between logs. It gets all the events names from all logs, not divided by logs like in Algorithm 1. Afterwards, the algorithm use Part-Of-Speech Tagger functionality from the Stanford Natural Language Processing Group [TOUTANOVA *et al.*, 2003] to tag the names. The Stanford part of speech tagger parses a phrase identifying the words' grammatical classes. To simplify the natural language processing, the algorithm looks for the first verb and the first noun from each event name, since best practices in modelling process is to use activities labels composed of one verb and one object (verb-object style) [MENDLING *et al.*, 2010].

Algorithm 2 – Tag event names.

Input: List of logs from Algorithm 1 (*listLogs*)

Output: Set of events name tagged by NLP tags (*listEventsTagged*)

```
1   Create list of string (listDifEvents) to get all unique names between logs
2   Foreach log log in listLogs do
3   |   Foreach event evt in listLogs do
4   |   Variable eventName = Call listLogs.get(log).getEvents().get(evt) function
5   |   If listDifEvents does not contain eventName
6   |   |   Add eventName to listDifEvents

7   Create list of NlpTag model (listEventsTagged) to get the events names tagged
8   Foreach eventName in listDifEvents do
9   |   Variable eventByList = eventName
10  |   NlpTag eventTagged = Call AccessStanford.phraseParse(eventByList)
11  |   Add eventTagged to listEventsTagged
12  Return listEventsTagged
```

Algorithm 2 – Tag event names.

Algorithm 2 calls the Algorithm 3 to parse the event name and to receive the part-of-speech tag. The parser from Algorithm 3 can read various forms of plain text input and can output various analysis formats, including part-of-speech tagged text, phrase structure trees, and a grammatical relations (typed dependency) format [TOUTANOVA et al., 2003]. The Tree is generated by the parse functionality from the library, which the text's words are leaves with its respectively grammatical tags. Therefore, the algorithm runs over the tree structure to get the tag for each word in the event's names searching for verbs and nouns.

Algorithm 3 – Stanford phrase parse.

Input: Event name from Algorithm 2 (*eventByList*)

Output: Event name tagged by NLP tags (*eventWord*)

```
1   Function phrase parse using Stanford library access
2   Initialize a variable lexicalized parser from Stanford library (parser)
3   Create a NlpTag object (eventWord) to get the events verbs and nouns
4   Initialize a variable Tree from Stanford library (tree) = parser.parse(eventByList)
5   List of Tree leaves = Call tree.getLeaves() function
6   Foreach leaf in leaves do
```

```
7    |  Variable gramaticalClass = leaf.parent(tree).label().value()
8    |  If gramaticalClass is a noun
9    |  |  Set eventWord  noun to leaf.label().value()
10   |  If gramaticalClass is a verb
11   |  |  Set eventWord  verb to leaf.label().value()
12   Return eventWord
```

Algorithm 3 – Stanford phrase parse.

The Algorithm 4 has the goal to find events' names that are synonyms of others events' names. This algorithm runs over the list of events tagged from Algorithm 2 getting the events and its grammatical classes to access synonym functionally from WordNet. Therefore, to be able to find synonyms, the algorithm use JWI library. The JWI (MIT Java WordNet Interface) is a Java library for interfacing with WordNet, which is simple to use and it does not require external libraries or files to run [FINLAYSON, 2014].

The algorithm runs over the list from Algorithm 2 (list of events tagged).

Algorithm 4 – Get events synonyms.

```
Input: List of events tagged from Algorithm 2 (listEventsTagged)
Output: Set of aspects (listAspects)
1    Create a list of list of events (listCloneEvents)
2    Foreach event evt in listEventsTagged do
3    |  Variable verb = evt.getVerb()
4    |  Get synonyms verbSyns = Call JWI.getSynonyms(verb, verb.getTag()) function
5    |  Create list of events synonyms (listSynEvents)
6    |  Foreach event evt2 in listEventsTagged do
7    |  |  Variable verb2 = evt2.getVerb()
8    |  |  If verbSyns contains verb2
9    |  |  |  Variable noun = evt.getNoun()
10   |  |  |  Get nounSyns = Call JWI.getSynonyms(noun, noun.getTag()) function
11   |  |  |  Variable noun2 = evt2.getNoun()
12   |  |  |  If nounSyns contains noun2
13   |  |  |  |  Variable firstEventName = evt.getEventName()
```

```
14  | | | | Variable secondEventName = evt2.getEventName()
15  | | | | Add firstEventName and secondEventName to listSynEvents
16  | Add listSynEvents to listCloneEvents
17  Return listCloneEvents
```

Algorithm 4 – Get events synonyms.

The Algorithm 5 consolidates the events found as synonyms and also finds the events with the same name. In the Algorithm 4, the search for synonyms ignores the word itself. For example, the synonyms for the verb “print” are [“publish”, “impress”], not including the word “print”. Therefore, events with the same words will not be found in the synonym algorithm, but they are present in the list from the Algorithm 1 when it was found the unique events by log. Therefore, the Algorithm 5 replace in the list of events from Algorithm 1 all occurrences from one of the events present in the synonyms list by one of the synonyms. Afterwards, the algorithm recognizes, among the logs, which events are the same, *i.e.*, it creates a list of events that exist in at least two different logs. The creation of a new list of events uses the TreeSet java class, which facilitates the comparison between elements.

Algorithm 5 – Consolidate events.

Input: List from Algorithm 4 (*listCloneEvents*) and Algorithm 1 (*listLog*)

Output: Set of aspects (*listAspects*)

```
1   Foreach event evtClone in listCloneEvents do
2   | Variable eventNameChosen = evtClone.getName()
3   | Variable evtSynon = evtClone.getSynonyms()
4   | Foreach log lg in listLog do
5   | | List of events by log (listEventsByLog) = lg.getEvents()
6   | | Foreach event evt in listEventsByLog do
7   | | | If evtSynon contains evt
8   | | | | Remove evt from listEventsByLog
9   | | | | Add eventNameChosen to listEventsByLog

10  Create a list of aspects (listAspects)
11  Create a list of all events (listallEvents)
```

```
12 Foreach log lg in listLog do
13 |  Foreach event evt in lg.getEvents() do
14 |  |  Add evt to listallEvents

15 Create a TreeSet with a comparator of events (allEventsSet)
16 Foreach event evt in listallEvents do
17 |  If evt is not added in allEventsSet
18 |  |  Add evt to listAspects
19 Return listAspects
```

Algorithm 5 – Consolidate events.

The output of this step (Algorithm 2, Algorithm 4 and Algorithm 5) is a list of aspects candidates.

4.2.3. Apply Cluster Approach

The cluster approach was inspired by the plugin Pattern Abstractions in ProM. ProM is an open-source framework for implementing process mining tools in a standard environment. It also provides process mining algorithms, which includes the Pattern Abstraction algorithm [VERBEEK, 2010]. It provides a functionality to automatically discover significant abstractions of the activities and pre-process the log using those abstractions. The ProM tool was chosen because it is the only framework with a large list of algorithms available to use. It integrates the functionality of several existing process mining tools and provides additional plug-ins [VAN DONGEN *et al.*, 2005]. Most of the other tools for process mining are not free and the user does not have the choice of which algorithm to use. It is also the official framework from processmining.org. In the ProM framework, it only was found one algorithm of pattern abstractions (Pattern Abstraction plugin) including the plugins that needs to be installed from the process mining package manager by the date of this dissertation (September, 2015). The Pattern Abstraction plugin is the implementation of the work of Bose *et al.* [2009].

Since this plugin groups activities to make the process more abstract, the method only does the functionality of grouping, it does not have the abstraction functionality. So, after the aspects identification by the clone approach, the method seeks to group the same aspects if they were executed in a sequence. For example, if the clone method finds three aspects (A, B and C), the cluster approach will verify if they were executed in a sequence in a certain number of cases.

The method receives as input the frequency it is supposed to group. That is, if the user wants to group aspects that are executed in a sequence above 70%, the approach verifies if aspect A and aspect B are executed in sequence in 70% of the cases. If it is true, then the method suggests grouping them. To calculate the frequency of each aspects found in Algorithm 5, it is necessary to build a linked list of this aspects with each one of them have a list of events that followed. The Algorithm 6 describes the logic used by the method to build this linked list.

Algorithm 6 – Building linked list

Input: List of logs from Algorithm 1 (*listXLogs*) and list of aspects (*listAspects*)

Output: List of aspects with frequency count (*listAspects*)

```
1   Foreach log xlog in listXLogs do
2   | Create an Iterator of XTrace traceIter = xLog.iterator()
3   | While traceIter has next
4   | | Create an Iterator of XEvent eventIter = xTrace.iterator()
5   | | | While eventIter has next
6   | | | | Create a XEvent variable xEvent = eventIter.next()
7   | | | | Event name = xEvent.getAttributes().get("concept:name")
8   | | | | If listAspects contains name
9   | | | | | Variable nextEventIdex = eventIter.nextIndex()
10  | | | | | Create a XEvent variable nextXEvent = xTrace.get(nextEventIdex)
11  | | | | | Event nextName = nextXEvent.getAttributes().get("concept:name")
12  | | | | | Create indexAspect = listAspects.indexOf(name)
13  | | | | | Create List eventsSeq = listAspects.get(indexAspect).getEventsSeq()
14  | | | | | If eventsSeq contains nextName
15  | | | | | | Variable count = eventsSeq.get(nexName).getCount()
16  | | | | | | count = count  +1
17  | | | | | | eventsSeq.get(nexName).setCount(count)
18  | | | | | Else
19  | | | | | | eventsSeq.get(nexName).setCount(1)
20  | | | | | listAspects.get(indexAspect).setEventsSeq(nexName)
```

Algorithm 6 – Building linked list

Algorithm 7 describes the logic to group the aspects by its frequency. The algorithm runs over the list of aspects received from Algorithm 5 recursively. It adds to each aspect from the list a list of aspects that were executed in sequence after itself.

Algorithm 7 – Clustering aspects

Input: List of aspects from Algorithm 5 (*listAspects*) and percentage *perc*

Output: Set of aspects (*listClusterAspects*)

```
1  Variable numberTraces = 0
2  Foreach log lg in lisXtLog do
3  |  numberTraces = numberTraces + lg.size()
4  Variable frequency freq = perc * numberTraces / 100

5  Create linked list of cluster (listClusterAspects)
6  Foreach aspect aspect in listAspects do
7  |  Variable next events eventsAfter = aspect.getEventsSeq()
8  |  Foreach event evt in eventsAfter do
9  |  |  If listAspects contains evt
10 |  |  |  If evt.getCount() >= freq
11 |  |  |  |  Create a list of all next aspects (listnextAspects)
12 |  |  |  |  Add aspect to listnextAspects
13 |  |  |  |  Add evt to listnextAspects
14 |  |  |  |  Call recursive function getAspectsSeq(listAspects,listnextAspects,evt,freq)
15 |  |  |  |  Add listnextAspects to listClusterAspects
16 Return listClusterAspects

17 Function getAspectsSeq(listAspects,listnextAspects,evt,freq)
18 If listAspects contains evt
19 |  Create list eventsAfter = listAspects.get(evt).getEventsSeq()
20 |  Foreach event evt2 in eventsAfter do
21 |  |  If listAspects contains evt2
22 |  |  |  If evt2.getCount() >= freq
23 |  |  |  |  If listnextAspects does not contains evt2
24 |  |  |  |  |  Add evt2 to listnextAspects
```

```
25  | | | | | Call recursively getAspectsSeq(listAspects,listnextAspects,evt2,freq)
```

Algorithm 7 – Clustering aspects

The output of this approach is a list of groups of aspects candidates. The output from the method is two lists. The first list contains the aspects candidates from clone approach, and other list presents the same aspects but grouped if their frequency is bigger than the given threshold.

4.2.4. Example of the Method Application

To exemplify the method, consider two logs from the processes described in Figure 11 and Figure 12. Both processes are related to a school department office from an university.

The process from Figure 11 represents the steps required for a student to get his registration. The process begins when the student fills a form indicating what he requires – in this case, he will require the school registration. The secretary, who works at the department office, prints the declaration of school registration. Afterwards, it gets the declaration stamp with the school logo and gets the signature of the responsible for the department. The final step would be to lay up the form in the student's folder.

The process from Figure 12 represents the steps to cancel de school registration. It starts when the student fills the same form as the process to require school registration declaration, but now requiring the registration cancelation. The director from the school analyzes the request and grants the cancellation of school registration. The secretary cancels the registration in the school system and files the form in the student's application folder.

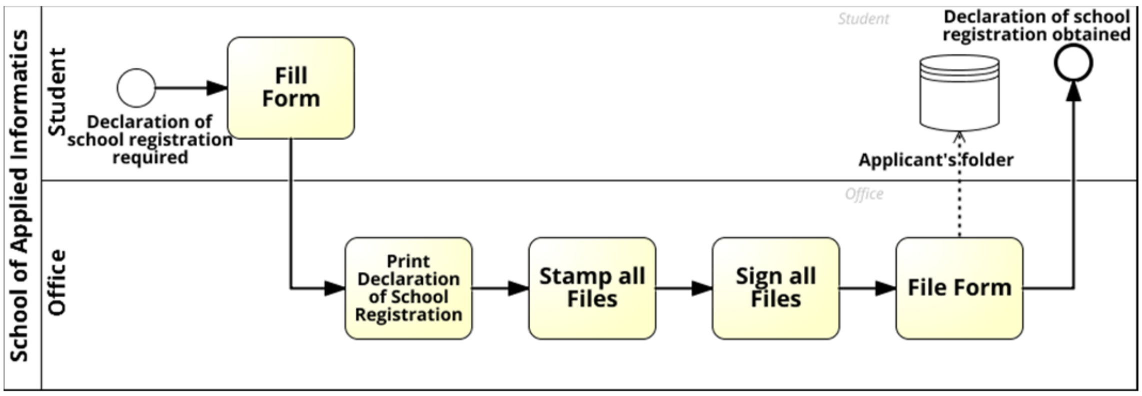

School of Applied Informatics
Student
Office
Declaration of school registration required
Fill Form
Student
Declaration of school registration obtained
Applicant's folder
Office
Print Declaration of School Registration
Stamp all Files
Sign all Files
File Form

Figure 11 - Declaration of school registration process.

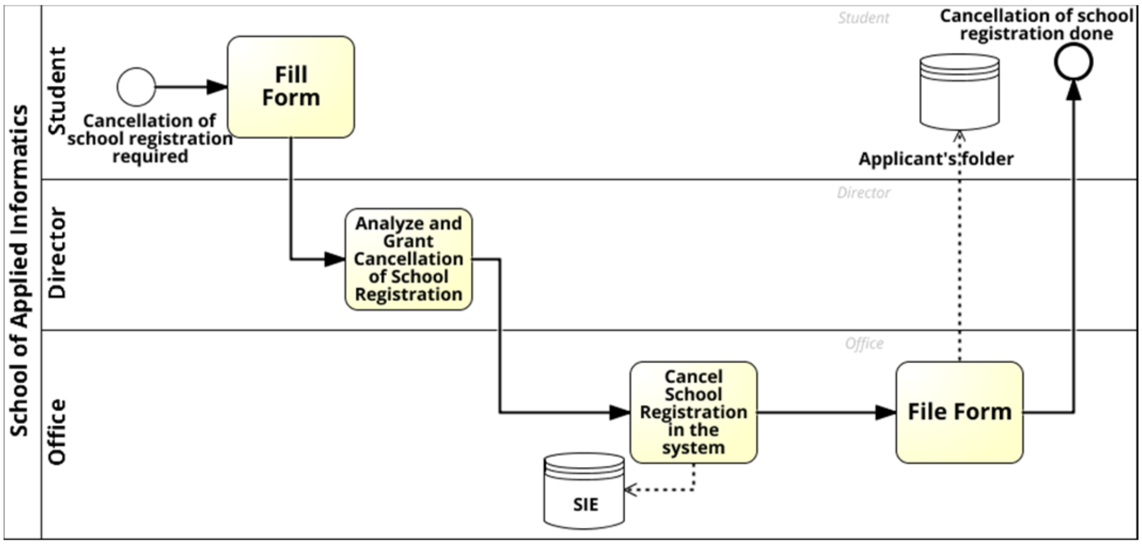

School of Applied Informatics
Student
Director
Office
Student
Cancellation of school registration required
Fill Form
Cancellation of school registration done
Applicant's folder
Director
Analyze and Grant Cancellation of School Registration
Office
Cancel School Registration in the system
File Form
SIE

Figure 12 – Cancellation of school registration process.

The aspect identification method, applied on these two processes, first, reads and loads the logs. After loading the logs of both processes from the execution of step 4.2.1 (Extract Process Elements), the method starts the aspects identification by the clone approach. Figure 13 illustrate a snapshot from the event log of the declaration of school registration process.

```
<trace>
    <string key="concept:name" value="12"/>
    <event>
        <string key="concept:name" value="Declaration of  school registration  required"/>
        <string key="lifecycle:transition" value="complete"/>
        <string key="org:resource" value="NOT_SET"/>
        <date key="time:timestamp" value="2015-08-06T08:00:00.000-03:00"/>
        <string key="Activity" value="Declaration of  school registration  required"/>
    </event>
    <event>
        <string key="concept:name" value="Fill  Form"/>
        <string key="lifecycle:transition" value="complete"/>
        <string key="org:resource" value="NOT_SET"/>
        <date key="time:timestamp" value="2015-08-06T08:00:04.000-03:00"/>
        <string key="Activity" value="Fill  Form"/>
    </event>
    <event>
        <string key="concept:name" value="Print Declaration of School Registration"/>
        <string key="lifecycle:transition" value="complete"/>
        <string key="org:resource" value="NOT_SET"/>
        <date key="time:timestamp" value="2015-08-06T08:00:07.000-03:00"/>
        <string key="Activity" value="Print Declaration of School Registration"/>
    </event>
    <event>
        <string key="concept:name" value="Stamp all Files"/>
        <string key="lifecycle:transition" value="complete"/>
        <string key="org:resource" value="NOT_SET"/>
        <date key="time:timestamp" value="2015-08-06T08:00:10.000-03:00"/>
        <string key="Activity" value="Stamp all Files"/>
    </event>
```

Figure 13 – Snapshot of the declaration of school registration process.

The second step from the method is step 4.2.2 (Apply Clone Approach), which runs through the structure created in step one and applies the Algorithm 2. Algorithm 2 tags the events names loaded with noun or verbs tags. For example, the event “Fill Form” would be tagged with “Fill” as verb and “Form” as noun. The algorithm checks every unique event name, in such a way, it will not do the same task to the same event name. That is, the algorithm creates a unique list with the events names and it would only tag the event “Fill Form” once, and not twice since the same event appears in both processes. The next step is the Algorithm 4, which seeks the events in the list from Algorithm 2 that can be synonyms from other events. This algorithm considers the grammatical classes in which the events were tagged. For example, the algorithm checks in the WordNet if the other words tagged as verbs (“Print”, “Stamp”, “Sign”, “File”, “Analyze”, “Cancel”) are synonym of the verb “Fill”. If it finds any match, the algorithm does the same to the nouns. If it does not find any match for the verb, the method did not find an event if the same meaning as “Fill Form”. In the process from Figure 11 and Figure 12, the method would not find any events meaning the same action. This means that the method would not replace the event name of one of the found synonyms. However, the Algorithm 5 consolidates the “clone” events by looking for the events with the same name. For example, the event “Fill Form” from process requires declaration of school registration is the same of event “Fill Form” from process to cancel the school registration. In this algorithm, the method would find the events “Fill Form” and “File Form” as aspects candidates. They are in the end and in the final of both processes.

The final step of the method is Apply Cluster Approach, which seeks to group candidate aspects. The input of this step is the output of the previous step (from the clone approach). This step would receive a percentage of frequency to group by this frequency. In the example presented, let’s consider 70% as the threshold and that the log size (total number of traces) is 2000. If any aspects are executed in sequence equal or higher than 1400 (70% * 2000) times, this approach should recommend to group the candidates. In the example, this step would check if the “Fill Form” and “File Form” were executed in sequence in logs traces adding up to 1400 times. The final results of the method in this example would be the suggestion of two aspects, “Fill Form” and “File Form” without suggestion of groups. Then, the specialist would decide what to do with the suggestion. If he would implement them as aspects or not.

5. Case Study

5.1. Scientific Methodology Approach

The analysis of the literature reports the extent, type, and nature of current organizational problems. These problems are the basis to formulate a research problem. The definition of research problem is a phenomenon which the researcher wishes to explain or predict. The analysis of the literature also reports where there are gaps of knowledge about a particular problem. This assists in identifying an important research question. This dissertation research question is: “How to identify automatically aspects in business processes?”.

After the specification of the research question, the next step is to plan the actions to answer the question. The action plan is called research design. Research design is the planning for the collection, measurement, and analysis of data. All research design types include observations, induction and deduction [RECKER, 2012].

The research design method suitable for this work according data, risks, theory, feasibility and instrumentation is the qualitative method. The qualitative method includes procedures which consist of research methods such as case study, ethnography or phenomenology. It is designed to assist researchers in understanding phenomena in context. Case study is the most popular form of qualitative methods and well-established published approach to research in information systems research and other social sciences, particularly business management. A case study is commonly used to investigate a contemporary phenomenon within its real-life context [RECKER, 2012].

A proof of concept is a demonstration, the purpose of which is to verify that certain concepts or theories have the potential for real-world application. Therefore, it is a prototype that is designed to determine feasibility, but does not represent deliverables [RECKER, 2012].

The scientific methodology approach used in this work is composed by 4 stages. In the first stage, the research problem and the hypothesis were defined, and the solution approach was designed. At the next stage, a theoretical background research was studied and described, looking for existing techniques to be compared or inspiration to be used in the proposed solution. In the following step, the solution was implemented. In the third stage, a proof of concept was performed to do an initial evaluation of the method.

The evaluation corresponded to a comparison of results from an example of the literature. That is, the processes chosen were the same used by Tavares and Marinho [2014] to check if the method identifies the same aspects identified by the experts in [TAVARES et al., 2014]. The fourth stage was the real life events case study. This case study was aimed to evaluate the proposed method using real life events logs and experts to access the results of the method execution over them. The results from both case studies (proof of concept and real life case study) are presented in the following subsections.

5.2. Proof of Concept

For the proof of concept, it was used the business process models from an administrative department of a university. The department is part of the Center of Sciences and Technology of the University of the State of Rio de Janeiro (UNIRIO). The processes correspond to services provided to students. They describe the steps that compose the interaction among actors (student, office, school director), for example, to get a second call of a test, to break discipline requirements, to get transferred in or out of the school, etc.

The processes were already modeled, however the method proposed receives logs as input, so it was used a tool (BIMP simulator - http://bimp.cs.ut.ee/) to simulate them, *i.e.*, to generate synthetic logs corresponding to that processes' models. The two processes chosen were the same used by Tavares and Marinho [2014] in order to be able to make a simple initial evaluation – comparison of aspects found by the method against the ones previously found by Tavares and Marinho in their work. However, the process models simulated was remodeled to be an acceptable input for the simulation tool. It also has a different modeling, which it was used to demonstrate an example of different abstractions levels. Tavares et al. [2014] propose improvements to the business process modeling notation from Cappelli et al. [2010] that uses the aspect orientation. In their work, they identify manually aspect from two processes to exemplify their proposal. Even though it is not the same process model, it is the same process by which the aspects identified should be the same.

The first process represents how a student obtains a discipline program. A discipline program presents the content of the discipline including the bibliographical references, themes and topics to be developed. The second process represents how the student can require the school record. Both processes represent the path the student has to follow to obtain those services at this university department. Figure 14 and Figure 15 illustrate the two processes.

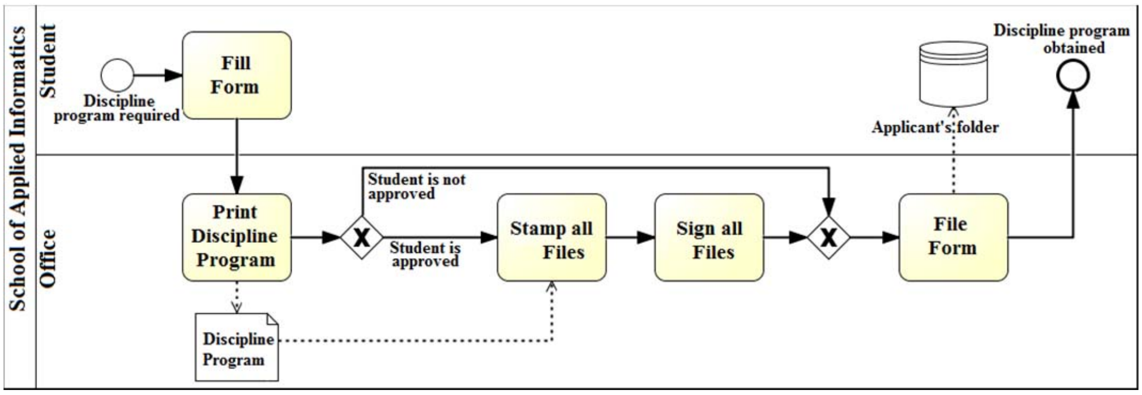


Figure 14 – Require discipline program process.

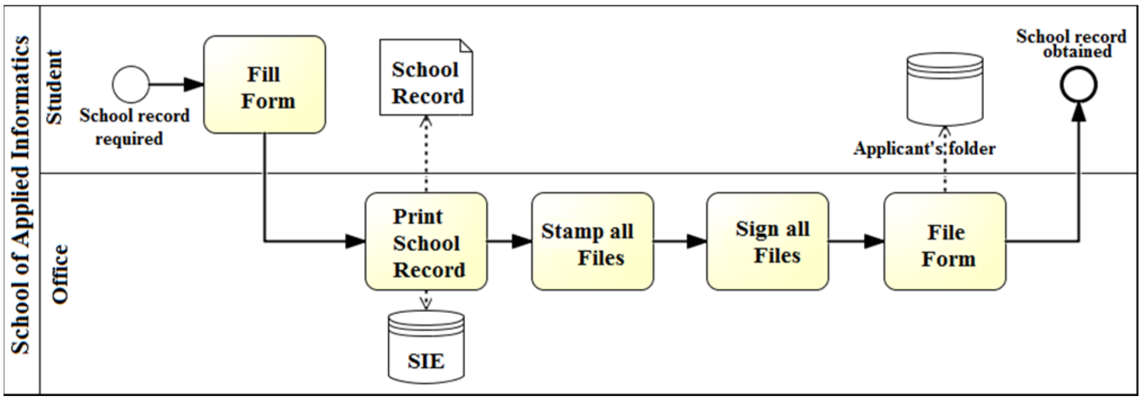


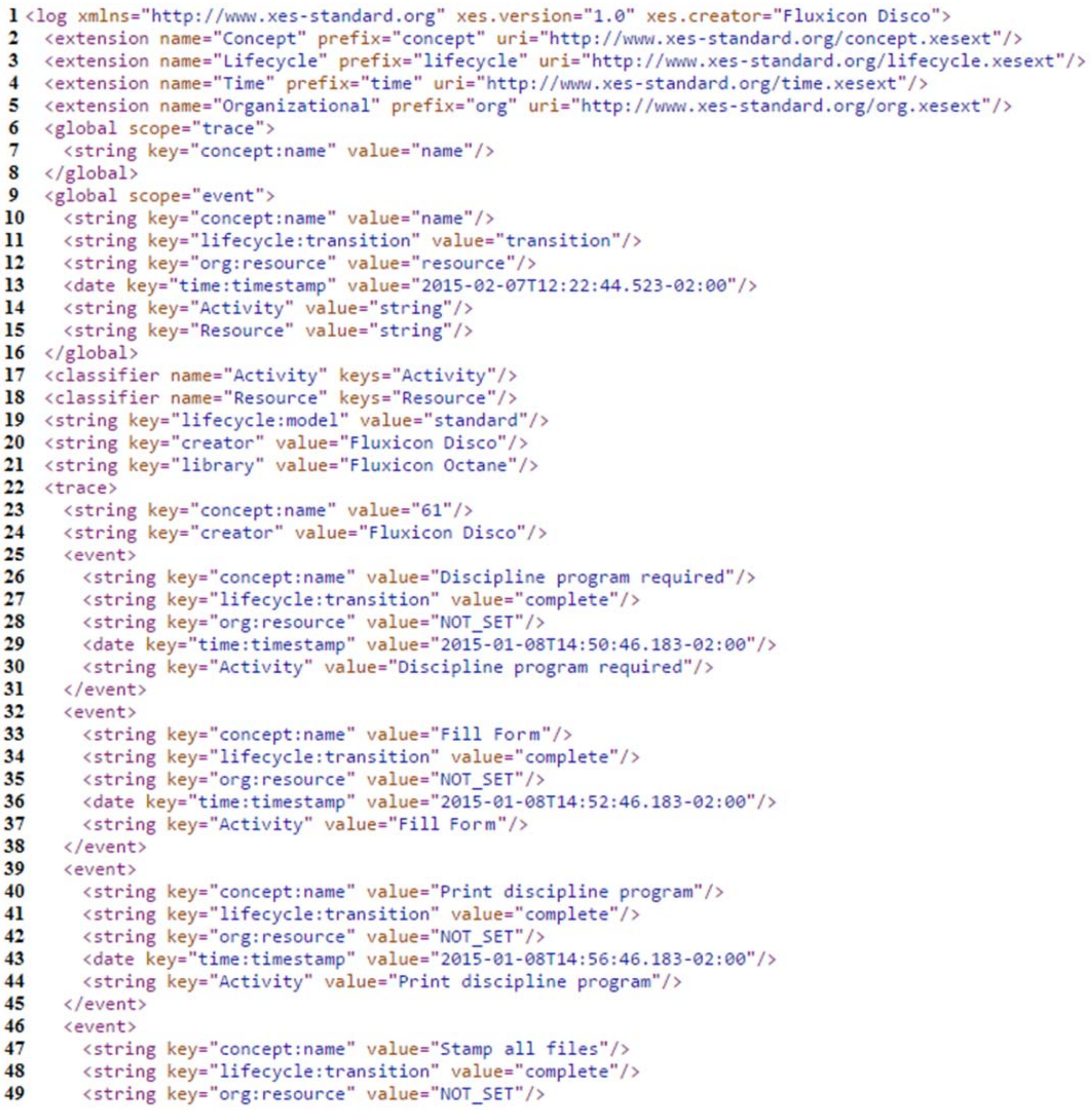
Figure 15 – Require school record process.

Figure 14 represents the process to require the discipline program. The discipline program process starts when a student fills the form. In the next step, the secretary prints the discipline program. Afterwards, the secretary checks if the student is approved in the required discipline. If the student is approved, the secretary stamp and sign all the files. If the student is not approved, the secretary skips the stamp and sign activities. Finally, the secretary files the form in the students' application folder.

Figure 15 represents the process to require the school record. This process is very similar to the discipline program process. It also begins with the student filling the form. The secretary prints the school record, stamp and signs all the files, and, finally, she files the form.

It was used a log simulator (BIMP Simulator [DUMAS *et al.*, 2013] [SIGNAVIO, 2014]) to create a synthetic log to test the method implementation. BIMP Simulator is a research prototype available as a part of the modeling platform of the BPM Academic Initiative. This tool allows simulating complex real-world business processes in large-scale scenarios. The system was designed and implemented by Dumas *et al.* at the University of Tartu [SIGNAVIO, 2014]. This simulator takes as input a BPMN process model in XML format produced by other process modeling tools such as Signavio Process Editor or OpenText Provision (format .bpmn or .vsdx) [DUMAS *et al.*, 2013]. This simulator was chosen because it has as input a BPMN process models and the two processes were modeled with the Signavio tool, which is possible to export the process as .bpmn for the simulator.

Figure 16 illustrates an extract of the log generated. The log has 9170 entries. Tag log (line 1) represents the log itself, whereupon it contains the tags trace that contains the tags events. Also, there are the tags of characterization of the log, *e.g.*, the tag global defines constraints, *i.e.*, what information each elements of the log must have. In Figure 16, there are two global definitions, one to the trace scope (lines 6 to 8) and other to the event scope (lines 9 to 16). The global definition of trace is requiring that each trace in the log contains a name. The global definition of event is requiring that each event in the log contains a name, a lifecycle transition, resource, timestamp and activity. After the tags of the log characterization, there are the traces tags that contain the events tags. In Figure 16, there is a tag trace with the attribute name and creator which contains 4 events tags (lines 22 and onwards). The first event tag contains “discipline program required” as value from the name field, and it has the others attributes (line 25).



```
<log xmlns="http://www.xes-standard.org" xes.version="1.0" xes.creator="Fluxicon Disco">
  <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
  <extension name="Lifecycle" prefix="lifecycle" uri="http://www.xes-standard.org/lifecycle.xesext"/>
  <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
  <extension name="Organizational" prefix="org" uri="http://www.xes-standard.org/org.xesext"/>
  <global scope="trace">
    <string key="concept:name" value="name"/>
  </global>
  <global scope="event">
    <string key="concept:name" value="name"/>
    <string key="lifecycle:transition" value="transition"/>
    <string key="org:resource" value="resource"/>
    <date key="time:timestamp" value="2015-02-07T12:22:44.523-02:00"/>
    <string key="Activity" value="string"/>
    <string key="Resource" value="string"/>
  </global>
  <classifier name="Activity" keys="Activity"/>
  <classifier name="Resource" keys="Resource"/>
  <string key="lifecycle:model" value="standard"/>
  <string key="creator" value="Fluxicon Disco"/>
  <string key="library" value="Fluxicon Octane"/>
  <trace>
    <string key="concept:name" value="61"/>
    <string key="creator" value="Fluxicon Disco"/>
    <event>
      <string key="concept:name" value="Discipline program required"/>
      <string key="lifecycle:transition" value="complete"/>
      <string key="org:resource" value="NOT_SET"/>
      <date key="time:timestamp" value="2015-01-08T14:50:46.183-02:00"/>
      <string key="Activity" value="Discipline program required"/>
    </event>
    <event>
      <string key="concept:name" value="Fill Form"/>
      <string key="lifecycle:transition" value="complete"/>
      <string key="org:resource" value="NOT_SET"/>
      <date key="time:timestamp" value="2015-01-08T14:52:46.183-02:00"/>
      <string key="Activity" value="Fill Form"/>
    </event>
    <event>
      <string key="concept:name" value="Print discipline program"/>
      <string key="lifecycle:transition" value="complete"/>
      <string key="org:resource" value="NOT_SET"/>
      <date key="time:timestamp" value="2015-01-08T14:56:46.183-02:00"/>
      <string key="Activity" value="Print discipline program"/>
    </event>
    <event>
      <string key="concept:name" value="Stamp all files"/>
      <string key="lifecycle:transition" value="complete"/>
      <string key="org:resource" value="NOT_SET"/>
```

Figure 16 – Generated log.

Based on the logs generated for these two processes by BIMP tool, the proposed method extracts the events from the log following the algorithms presented in Chapter 4. The output of this step is a list containing all the events by log. It is extracted all the events in the log and store by its log so that it can be known each event is from. The list of events by log: "Discipline program required", "File Form", "Discipline program obtained"; and from the other log: "School record required", "Fill Form", "Print School Record", "Stamp all Files", "Sign all files", "File Form".

The next step is the clone approach, which runs through the list searching for equals and synonyms events. In this case, this approach found four aspects candidates ("Fill Form", "Stamp all files", "Sign all files" and "File Form").

The final step from the method is to apply the cluster approach, which runs through the logs searching to group the aspects candidates found by the previous step (the clone approach). In this case, the threshold to seek the aspects that were executed in sequence was seventy percent. That is, the cluster approach search through the log checking if the four aspects (“Fill Form”, “Stamp all files”, “Sign all files” and “File Form”) were executed in sequence for more than seventy percent of the traces. Since the log size simulated had 200 traces each, the frequency of 70% is equivalent to an appearance of 280 ((200+200) * 70%) times. This approach found two cluster aspects. One group containing three aspects (“Stamp all files”, “Sign all files” and “File Form”) which can be grouped as one aspect, and the second group found contains two aspects (“Sign all files” and “File Form”) which can be grouped as one cluster aspect. If the threshold is one hundred percent, the method could still find the four aspects from the clone approach, but it did not find any cluster aspect. This results that both cluster aspects candidates found with a seventy percent frequency were not always executed in sequence. There were traces in the log, instances of the processes, where these aspects were not followed by one another.

The final results of the proposed method for this proof of concept using the business process of an administrative department of an university illustrated in Figure 14 and Figure 15 were four aspects. It also recommends grouping three or two of these aspects, but with a seventy percent frequency that they are executed in sequence.

These processes were chosen as the same from Tavares and Marinho work in order to compare the results of the proposed approach to their proposal. Tavares and Marinho [2014] used these two same processes as examples to propose improvements on the AO-BPM notation. Although it is the same process, they used different process models, their models from obtain the discipline program and require the school record were more detailed, *i.e.,* their models have more activities describing the process. The processes models used in this work were the processes from a different project, which all processes from the school office was modeled. Even though the process models are different, it was the same process used by Tavares and Marinho. Figure 17 illustrates the require discipline program process model used by Tavares and Marinho. Making a comparison of the process used in the proof of concept (Figure 14) with the process used by Tavares *et al.* [2014], it can be seen that the process in Figure 17 has more details. These details are a more specific activity description like the activity “Wait student to pick up document”. It also has as detail the use of the data object “Discipline program”. Finally, it has the details of an extra actor (Director), which in the process from Figure 14 was implicit and subsumed in the office action. The same level of details are also presented in the require school record process.

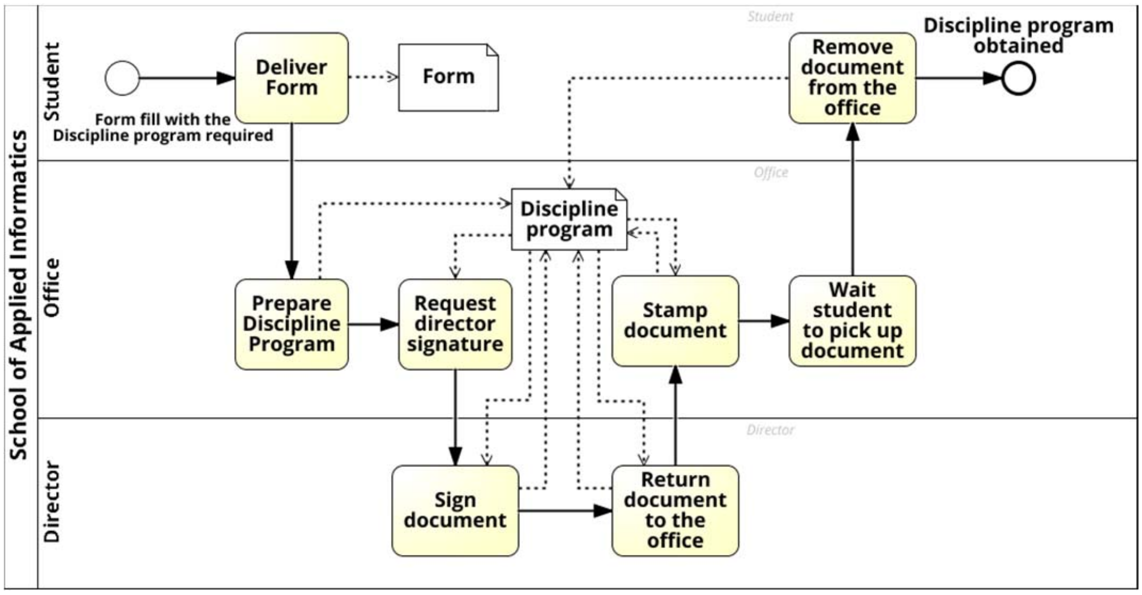

School of Applied Informatics
Student
Office
Director
Form fill with the Discipline program required
Deliver Form
Form
Remove document from the office
Discipline program obtained
Discipline program
Prepare Discipline Program
Request director signature
Stamp document
Wait student to pick up document
Sign document
Return document to the office

Figure 17 - Require discipline program process from Tavares et al. [2014]

In their work, they found seven aspects (“Deliver form”, “Request director signature”, “Stamp document”, “Sign document”, “Return document to the office”, “Remove document from the office”, “Wait student to pick up document”), wherein the aspects “Request director signature”, “Stamp document”, “Sign document”, “Return document to the office” are grouped as one aspect and “Remove document from the office”, “Wait student to pick up document” are also grouped in another aspect.

The aspects found in this proof of concept by the method proposed could correspond to the aspects modelled by Tavares and Marinho’s work [2014]. Doing a simple interpretation of the aspects identified by both works, it could claim that the aspect “Fill form” corresponds to the same activity “Deliver form” from Tavares and Marinho’s processes. The same with the group “Stamp all files”, “Sign all files” and “File Form” that might corresponds to “Request director signature”, “Stamp document”, “Sign document” and “Return document to the office”. Since the used process model did not have details about how the student get the required document, the method could not find the aspects corresponding to “Remove document from the office” and “Wait student to pick up document” from Tavares and Marinho’s work. The process models used in the proof of concept were not the same from Tavares and Marinho’s work because the difference from the same processes shows the abstraction level problem. The proposed method uses process logs as input; the process models in Figure 14 and Figure 15 are only to illustrate the process. It was used a tool to simulate the process to create a synthetic log, since it is a tool from a known project (BPM Academic Initiative) and the log is created randomly, the results can be considered positive.

This initial study shows that the method proposed in this work could help in aspect identification process. However, the details from the processes affect the results. That is, the level of abstraction from the processes should be the same this method work. For example, if it was used one process from Tavares and Marinho's work and one process from this work, the method wouldn't discover any aspects because the level of details from the processes are different. The evaluation of this method was qualitative, dependent of specialist opinion to check if the aspects found were accurate. However, this initial results show that our method could have been used by Tavares and Marinho's work to assist and to facilitate the aspects identification.

There are some limitations of this study, for example the process context. The aspects found can only be confirmed as aspect in the context from the loaded logs. It is possible that could have more or less aspects than the aspects found if you consider different context or processes. For example, maybe considering different process of the school office as the transfer of a student, the event "Sign all Files" is not present. This would result in the method not identifying it as an aspect. Therefore, the validity of the study is dependent of the context from the logs loaded.

This proof of concept has some threats to the validity of the method. The use of simulated logs of trivial processes rather than using actual logs of complex processes is one threat, but the next section shows the method using real-life logs. The last threat is the way activities labels are written. This proposal assumes activities labels written accordingly to good practices, such as verb plus noun. If the labels are not written in this format, the step from finding synonyms from the method will not be able to tag the words, resulting in not finding aspects or decreasing the method accuracy.

5.3. Real Life Events Case Study

This case study has the goal to evaluate the method in a real life scenario. It consists in executing the proposed method to identify aspects from a real life event logs. The results obtained with the method executed were then accessed by specialists in two stages. The first stage was to ask the interviewees to identify aspects from the process models generated from the logs. The second stage was to compare the aspects found by the specialist with the aspects found by the method application. The evaluation covers the specialist opinion about the results and the data collected from the observation over the interview. In the literature study, it was not found works which used events logs to identify aspects. It was not used precision and recall as evaluation because there is not a similar work to be used as an answer.

5.3.1. Event log pre-processing scenario

The real life events log used was from the Business Process Intelligence Challenge 2014 (http://www.win.tue.nl/bpi/2014/challenge). This data set represents the log information from the execution of ITIL processes in a Dutch bank. ITIL (Information Technology Infrastructure Library) is a set of practices for information technology service management, which focus on aligning IT services with the business needs [ARRAJ, 2010]. This events log was generated by the bank system logging theirs steps. It is not an events log generated by system created from business process models, *e.g.* systems created through BPEL (Business Process Execution Language). The method from this work can have as input any events logs since it follows the XES standard. Therefore, events logs generated by BPEL can be used in this method.

The data set contains four data sets, which it was loaded in one SGBD to be analyzed. The four data sets are the interactions records, incidents records, activities of the incidents records and the changes records. The relation between these sets is once the helpdesk service call is created, it is an interaction record. If this interaction occurs two times or more, it becomes an incident. The dataset activities of the incidents record provides the information of the actions executed of the incident created (name, date, assignment group – who did the task) and the change record is the dataset which records the change in a system service. The change record dataset was not used. A subset of this dataset was selected to be used in this case study. The first filter was to select only the interaction that has the category as incidents occurrences because it is the process with most occurrences from the whole dataset. The final filter was to select only the interactions with status marked as closed, which still continues as the most occurrences of the dataset. These filters were applied to unify data from the same process and to reduce the volume of data. A snapshot from the datasets is in Appendix IV. The dataset of interactions has 147005 records which contain 17 fields: Configuration Item Name

Affected, Configuration Item Type Affected, Configuration Item Subtype Affected, Service Component Affected, Interaction id, Status, Impact, Urgency, Priority, Category, Knowledge Document Number, Open Time, Close Time, Closure Code, First Call Resolution, Handle Time and Related Incident. The dataset of incidents has 46607 records with 28 fields. The fields are the same of interaction minus the first call resolution field and plus the fields: Reopen Time, Resolved Time, Alert Status, Number of Reassignments, Number of Related Interactions, Related Interaction, Number of Related Incidents, Number of Related Changes, Related Change, Configuration Item Name Caused By (the item which caused the service disruption), Configuration Item Type Caused By, Configuration Item Subtype Caused By. Finally, the incident activity dataset has 466738 records with 7 fields: Incident ID, Date Stamp, Incident Activity Number, Incident Activity Name, Assignment Group, Knowledge Document Number and Interaction ID. Table 3, Table 4 and Table 5 are a snapshot from the respectively dataset. In the SGBD, it was select all interactions that have the classification of incident and with status closed. Afterwards, to be sure that only have interactions that resulted in creating an incident, it was made an analysis checking the interaction by its ids with the incidents id. This last filter decrease error of interactions classified incorrectly and it also makes sure to use only incidents related to an incident. It was not necessary to use the log with all of the information in the method, thus the only information used after the filters was the fields presented in the incident activity dataset (Incident id, date stamp, incident activity name and assignment group). The method only needs the minimum information required by the XES format (case id, date stamp and activity name). One important observation of this dataset is the name of the activities. They do not represent the action taken by the company assignment group; they represent the action taken in the incident record in general. For example, the activity name "update" say that the record was updated, but do not say what information was updated. This lack of detail could be because the company did not want to make their processes information too public. Therefore, the log information released is a little abstract.

According to Reichert *et al.* [2015]: "Usually, there exists a multitude of variants of a particular process model, whereby each of these variants is valid in a specific scenario or in the context of a particular business objective (Lohrmann and Reichert 2012); *i.e.*, the configuration of a particular process variant depends on concrete requirements building the process context (Hallerbach *et al.* 2008b). Regarding release management, for example, we have identified more than twenty process variants depending on the considered product series, involving suppliers or development phases. Similar observations can be made with respect to the product creation process in the automotive domain for which dozens of variants exist. Thereby, each variant is assigned to a particular product type (*e.g.*, car, truck, or bus) with different organizational responsibilities and strategic goals, or varying in some other aspects.". Therefore, the divisions of incidents are variants of the helpdesk service as a whole, but it can be classified as a specific process by its service component.

The method proposed assumes the existence of at least two logs. To fulfill this assumption, the dataset was divided by service component to create specifics processes. That is, the incidents were divided by the service components they affect. The most common services are “WBS000263”, “WBS000073”, and “WBS000091”. Therefore, the whole log file represents a generic process, but it also contains specifics process for each one relates to one specific service component. Then, the preprocessing was to create separated log files for each one of the three most common services.

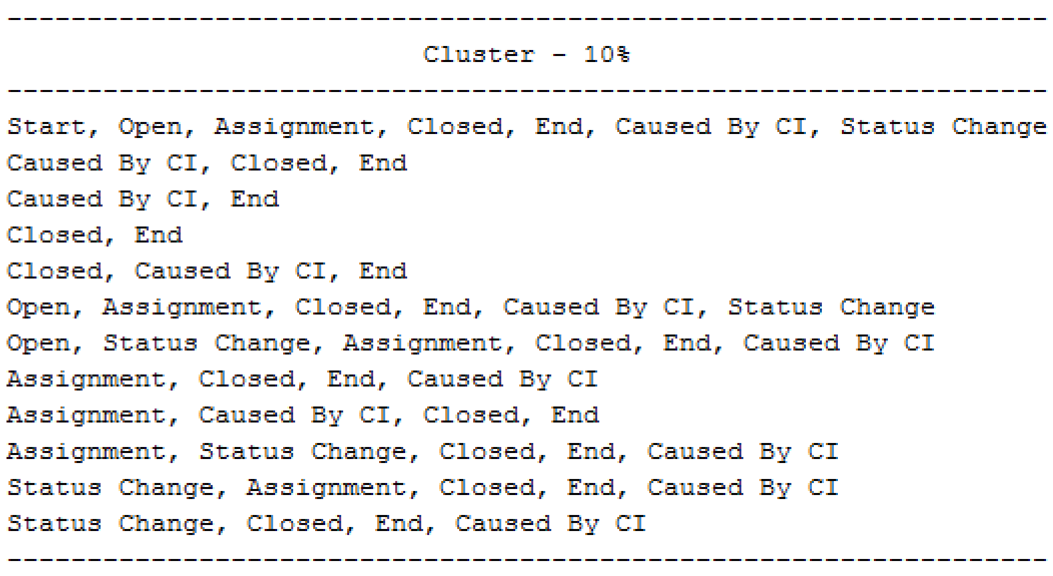
Beyond this division by service component, the process model generated was still too big to be understood and analyzed. The log is extensive and it has incorrect records, *i.e.*, incomplete traces or incidents store wrongly (noise). And since it is also based on the idea that the evaluation of the results would be with specialists, the logs were filtrated with only 5% of variants. That is, the processes model generated represents the most executed process, with only 5% of variants. Figure 21, Figure 22 and Figure 23 represents the model of the process for each service component. The models were extracted from the log using the Fuzzy miner algorithm [GÜNTHER *et al.*, 2007] using the Disco tool (https://fluxicon.com/disco/).

Disco was chosen this time instead of ProM because of its simplicity, usability and performance when dealing with huge logs [RUBIN *et al.*, 2014]. In the ProM framework the log from this case study took minutes to load, as the same file in Disco took seconds. However, the Disco tool does not give the option of choosing which algorithm to use unlike the ProM framework which provides a list of algorithms with multiples functionalities. Disco is a commercial product that uses the fuzzy miner algorithm to create a process model. The Fuzzy miner algorithm was the first to directly address a large number of activities. It also is recommend to use when you want to simplify the model in an interactive manner [GÜNTHER *et al.*, 2007]. The Disco user interface is also more friendly, being easiest to use the variants filter functionality to divide the log by service component.

After the filtering by record type (incident) with the category as closed and divided by the services component, the process of service “WBS000263” has ten activities, the process of “WBS000073” with nine activities and process of “WBS000091” with fourteen activities. The process model for “WBS000091” is presented in (Figure 21), the process model for “WBS000263” in (Figure 22), and the process model for “WBS000073” in (Figure 23).

5.3.2. Proposed method execution

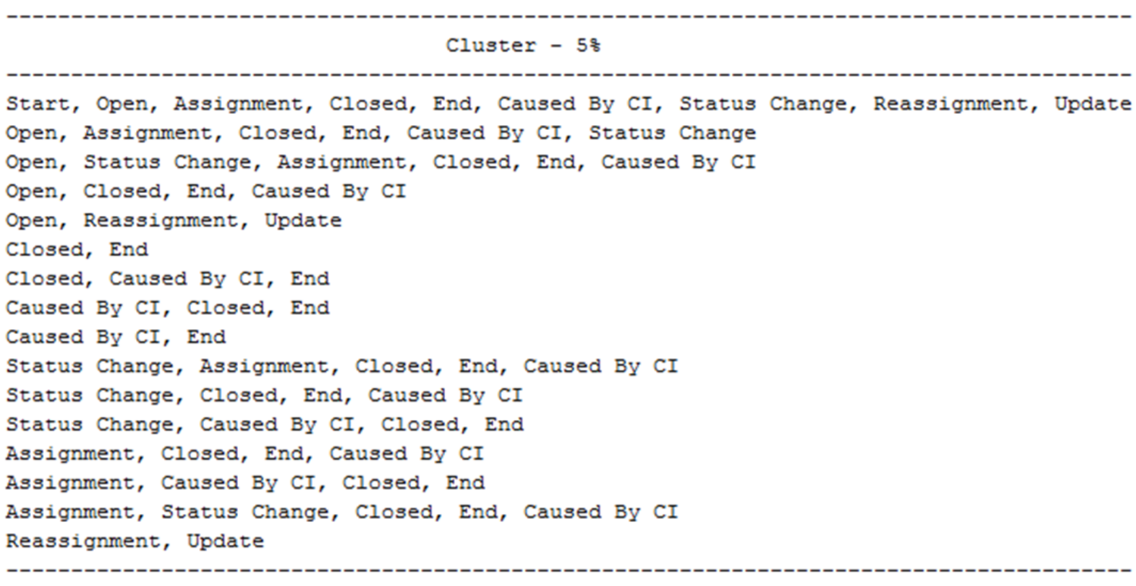
Based on the logs extracted for these three processes, the proposed method extracts the events from the log following the algorithms presented in Chapter 4. The output of the first step was a list containing all the 33 events. The next step is the clone approach, which runs through the list searching for equals and synonyms events. The method was implemented to find clone events corresponding the ones existing in all load processes, *i.e.*, a listed event has to occur in all three processes to be considered as an aspect. This feature can be configured by the user. This step found 7 aspects (“Start”, “Caused By CI”, “Closed”, “Open”, “Assignment”, “End” and “Status Change”). The cluster algorithm, considering a threshold of 70%, finds one group of aspects (“Start” and “Open”). This means that the initial event “Start” from the process and the activity “Open” are executed in sequence more than 70% of the traces of the log. If the threshold goes down to 10%, the cluster approach finds 12 possible groups. Figure 18 illustrates the console output of the 12 groups. It is important to observe that the method looks for events in the log that has name. This means that the candidate aspects “Start” and “End” are the initial and end event of the process model, and not an activity. This observation will be discussed in details in the evaluation.



```
-------------------------------------------------------------------
                           Cluster - 10%
-------------------------------------------------------------------
Start, Open, Assignment, Closed, End, Caused By CI, Status Change
Caused By CI, Closed, End
Caused By CI, End
Closed, End
Closed, Caused By CI, End
Open, Assignment, Closed, End, Caused By CI, Status Change
Open, Status Change, Assignment, Closed, End, Caused By CI
Assignment, Closed, End, Caused By CI
Assignment, Caused By CI, Closed, End
Assignment, Status Change, Closed, End, Caused By CI
Status Change, Assignment, Closed, End, Caused By CI
Status Change, Closed, End, Caused By CI
-------------------------------------------------------------------
```

Figure 18 – Output of aspects exclusively by cluster approach with 10% threshold.

The configuration of exclusively find aspects in all logs can be changed. If this feature is changed to identify in any logs, not being exclusively to all logs loaded, the clone approach finds 12 aspects (“Start”, “Open”, “Closed”, “Caused By CI”, “End”, “Status Change”, “Assignment”, “Reassignment”, “Update”, “Operator Update”, “Description Update” and “Resolved”). In this case, the cluster approach with 70% of threshold found one aspect groups (“Start” and “Open”). If the threshold would be 10%, the groups found are still the 12 found from Figure 18. However, if the threshold goes to 5%, the cluster approach finds 16 possible groups.



```
-----------------------------------------------------------------------------------------
                                  Cluster - 5%
-----------------------------------------------------------------------------------------
Start, Open, Assignment, Closed, End, Caused By CI, Status Change, Reassignment, Update
Open, Assignment, Closed, End, Caused By CI, Status Change
Open, Status Change, Assignment, Closed, End, Caused By CI
Open, Closed, End, Caused By CI
Open, Reassignment, Update
Closed, End
Closed, Caused By CI, End
Caused By CI, Closed, End
Caused By CI, End
Status Change, Assignment, Closed, End, Caused By CI
Status Change, Closed, End, Caused By CI
Status Change, Caused By CI, Closed, End
Assignment, Closed, End, Caused By CI
Assignment, Caused By CI, Closed, End
Assignment, Status Change, Closed, End, Caused By CI
Reassignment, Update
-----------------------------------------------------------------------------------------
```

Figure 19 – Output of aspects not exclusively by cluster approach with 5% threshold

The method result used to do the comparison evaluation is from the configuration with not exclusivity, that is, the identification of aspects is done in at least two logs and not limited in all logs. And the initial threshold would be 70% because it means that a significant number of times that the aspects were executed in sequence; it would be a relevant frequency. However, due to the size of this logs (3073 traces), the threshold used was 10%. This was the bigger percentage established for the method to find aspects groups besides the “Start” and “Open” group. This change was made because the process model does not show how frequent the path from one element to other element was done. Even though the logs only have 5% of variants, it is still a lot of different paths. Therefore, the limit to find groups of aspects executed in sequence can be low.

5.3.3. Evaluation by specialists

The evaluation for this real life log events is qualitative based on the experience of professionals with the business process management and aspects.

It was asked to all the interviewees to identify aspects from the process model generated from the logs, since to ask a human to identify from the logs files is unfeasible. The difference of identification from models and logs brings assessment bias to the evaluation. One bias is that the models not always show all the elements presented in the log. Another bias is that the model also does not show how many times sequence of events were done during a period of time. For example, the actions of grouping aspects made by the interviewees were basically identified by the semantic of the activities, and not by the frequency they were executed. Other bias can be the induction of not considering elements from the process model, for example, only one interviewee identified a gateway as a possible aspect; the others only payed attention to the activities.

The most complaint problem by the interviewees was to understand the process. Since the process model was generated from a process mining algorithm, the process model has some connections that the same process model by a human would not have. For example, after the activity “Closed” were some paths to the beginning of the process. Other problem was the name of some activities as events. For example, there are some activities that actually are events, but the log is recorded incorrectly; the activities “Resolved” and “Caused by CI” were one of this kind of problem reported by the interviewees.

5.3.4. Interviews

Three professionals were invited for this case study. Regarding the concept of aspect, all interviewees reported knowledge about it. They all reported knowledge of aspect in business process management, but only one of them knows aspect-oriented programming. The first step of the interviews was to explain the process. Then, the next step was to ask them to identify in the process model what they would consider aspect.

5.3.4.1. Interviewed #1

The first interviewed has 3 years of experience working with business process management and he did not have contact with aspect-oriented programming; however he is familiar with the concept of aspects in process models. He classified his knowledge in aspects as intermediate and he had a good knowledge from the log used because he had worked with it in the past. The first action of this interviewee was to look for activities that were present in all model processes. He only considered activities. His next step was to look for semantically similar activities. This interviewee in particular did not group aspects by its sequence, only by its semantic. For example, he identified the activities “Operator Update”, “Update” and “Description Update” as one aspect of update information. He was in doubt if the “Open” and “Closed” activities could be grouped as one aspect of the control of the beginning and ending from the processes. As the final results, this interviewee found 6 aspects, for which he grouped and named. The aspects are:

1) Opening – Activity “Open”
2) Closing – Activity “Closed”
3) Designation – Activities “Assignment”, “Reassignment” and “Update”.
4) Communication with client – Activities “Mail to Customer” and “Communication with Customer”.
5) Update status – Activity “Status Change”.
6) Update information – Activities “Update”, “Description Update” and “Operator Update”.

When asked about the aspects found by the method, the interviewee agreed with some of the results. He did not agree by the “Cause by CI” and “Resolved” because they are events and not activities. However, he did not consider this as a problem of the method, but a problem of the log that probably has a wrong information. He also observed that the activity “Update” is lacking an object to be with compliance of the good practices of verb plus object. He agreed with cluster approach of grouping by the frequency presented in Figure 18, but removing the activities “Caused by CI” and “Resolved”.

However, he expected that the method could had found the aspect communication with client (activities “Mail to Customer” and “Communication with Customer”). He also expected that the method have the functionality of grouping activities by its similarity and not only by frequency. Finally, he agreed with the “Start” event and “End” event as aspect and he recognizes that he did not pay attention to other elements but activities.

5.3.4.2. Interviewed #2

The second interviewee has 1 year experience working on business process management, but he has 6 months of experience with aspect-oriented programming. He had a 4 months of experience with aspects in process models. This interviewee classified his knowledge in aspects as basic. In comparison with the first interviewee, this person did not make assumptions that the activities “Assignment” and “Reassignment” are the same action. He afirmed that since he did not know the process very well he could not make this assumption. Since he had worked with aspect in programming, he only identified aspects that could be executed automatically, without human intervention. For example, he did not consider the activity “Open” as an aspect because it needs a manual behavior. He focused on the reuse heuristic to identify aspects. This interviewee found 3 aspects, they are:

1) Activity “Closed”
2) Activities “Update”, “Assignment” and “Status Change”
3) Activities “Update”, “Reassignment” and “Status Change”

This interviewee also complaint about parts of the process that do not make sense for him. When asked about the aspects found by the method, this interviewee also did not agree with the identification of the “Cause by CI” and “Resolved”. The researcher argued about why the activities could be manual and the interviewee agreed that the tasks could be made by human, but he pointed that he had the line of reasoning from the software engineering. After this conversation, the interviewee agreed that the aspects found by the method (“Start”, “Open”, “End”, “Operator Update”, “Description Update”) could be considered aspects. He also agreed with the group of aspects found by the frequency (Figure 18), but again, like the first interviewee without the activities

“Cause by CI” and “Resolved”. Even after the agreement of aspect could be manual activities, this interviewee did not consider to group activities by the semantic similarity (different than the first interviewee).

5.3.4.3. Interviewed #3

The third interviewee has 7 years working with business process management and he had no contact with aspect-oriented programming, but he had already worked with aspects in process models. This interviewee classified his knowledge in aspects as intermediate. This interviewee was in doubt about the level of abstraction, if he could assume that some activities are the same or if it could abstract an activity to include others activities. He did not know where to draw the line (limit) of the abstraction for reuse. For example, if he could consider the activity “Update” an abstraction of the activities “Operator Update” and “Description Update”. He argued that if he considers some abstractions to the limit, everything could be an aspect. However, he considered the activity “Assignment” the same as the activity “Reassignment”. This interviewed had work experience with aspects in process model, but in a different notation – EPC (Event-driven Process Chain) and not BPMN. Therefore, he observed a lack of information in the model (data objects as input/output of activities, business rules, data store, business requirements etc.). He also found difficult to understand the process. This was the first interviewee to consider elements apart from activities. This interviewee found 14 aspects, which are:

1) Activity “Open”
2) Activities “Assignment” and “Reassignment”
3) Activity “Operator Update”
4) Activity “Update”
5) Activity “Status Change”
6) Activity “Closed”
7) Activities “Communication with Customer” and “Mail to Customer”
8) Activity “Description Update”
9) Gateway before the activities “Communication with Customer” and “Mail to Customer”
10) Gateway after the activity “Closed” and before the end event
11) Group of activity “Closed” with gateway from item 10).

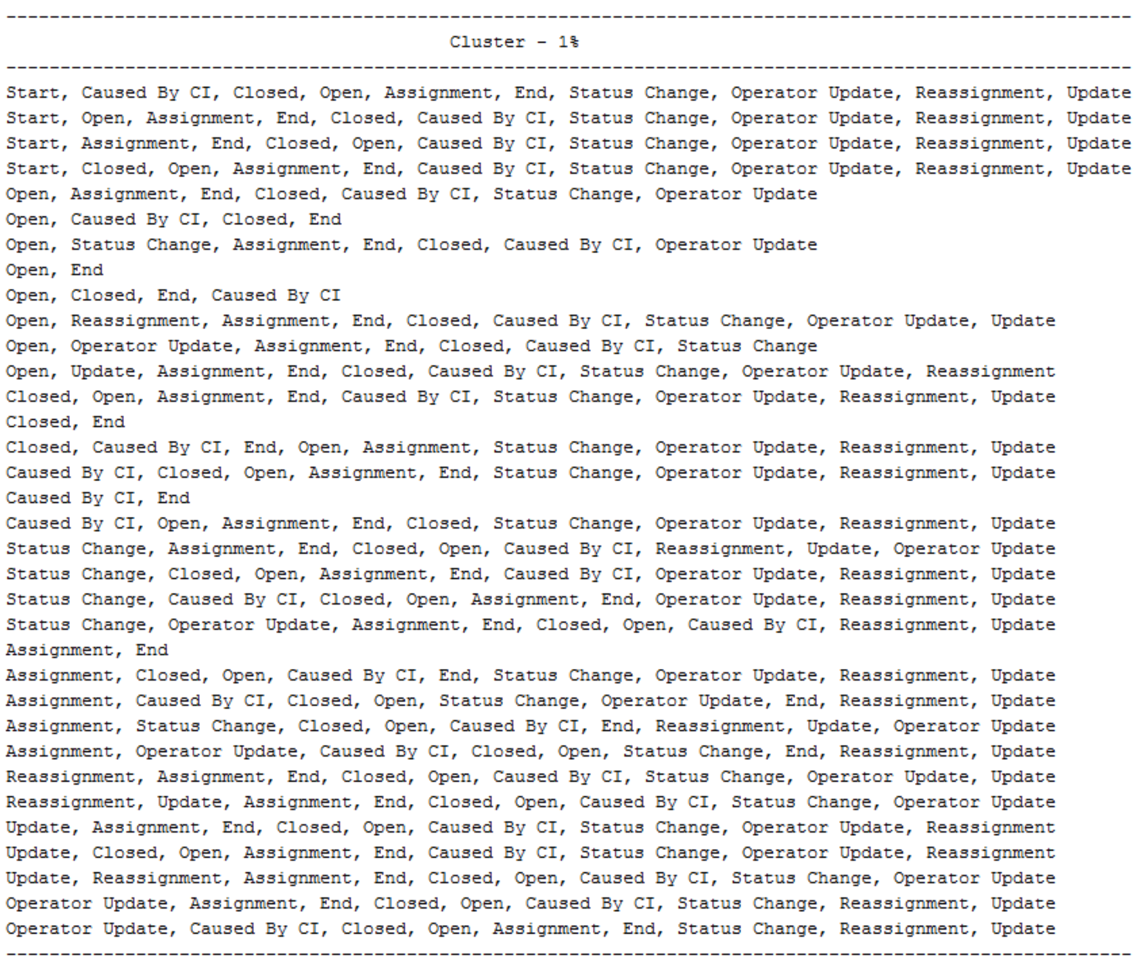
12) Group of activities “Status Change”, “Communication with Customer”, “Mail to Customer” with the gateway from item 9).
13) Group of activities “Open” and “Assignment”
14) Group of activities “Open”, “Assignment”, “Reassignment”, “Operator Update”, “Update” and “Description Update”

As the others interviewees, this third interviewee did not agree with the aspect “Caused by CI” and “Resolved” from the method. Another disagreement was the aspects from the gateways. And one doubt that this interviewed had was if the aspect of “Communication with Customer” and “Mail to Customer” could be considered the same activity. He was in doubt because he did not know how the process works enough to affirm if it is relevant that one of the communications is specific by mail and other activity can be by any other communication mean. He also agreed with the aspects found by the cluster approach from the method (Figure 18), but again without the activities “Caused by CI” and “Resolved”. This interviewee was the only one to identify groups of aspect by its sequence in execution, not only by its semantic similarity.

5.3.5. Discussion

All the interviewees shared some observations. The first point is that some parts of the process are hard to understand. The second one is that there are mistakes in the process model; two activities have names that mean events and not actions (“Resolved” and “Caused by CI”). Finally, they all agreed that it was the difficulty to know in which step to stop the abstraction to identify aspects. Two of the interviewees think that if one abstracts in some level the activity, it would become an aspect.

The most different results were the groups suggested by the cluster approach. Two of the interviewees grouped by its similarity, but the third one that grouped by its sequence had also different results. It can be inferred that the results are different because the process model do not show how frequent that path was taken. This information could be added to the process model, but it would not follow the guidelines of the BPMN notation. This way, the process model can induce to group aspects that were only taken one time in over one hundred traces (1% of frequency). Using a threshold of 10% to the cluster approach, the suggested aspects groups are more similar than the groups identified by the interviewees. It was this group of suggestion that was used in the case study (Figure 18). If the threshold had been 1%, the method would have found 34 suggestions of groups. Figure 20 illustrates the list of these 34 groups.



```
-----------------------------------------------------------------------------------------------------
                                        Cluster - 1%
-----------------------------------------------------------------------------------------------------
Start, Caused By CI, Closed, Open, Assignment, End, Status Change, Operator Update, Reassignment, Update
Start, Open, Assignment, End, Closed, Caused By CI, Status Change, Operator Update, Reassignment, Update
Start, Assignment, End, Closed, Open, Caused By CI, Status Change, Operator Update, Reassignment, Update
Start, Closed, Open, Assignment, End, Caused By CI, Status Change, Operator Update, Reassignment, Update
Open, Assignment, End, Closed, Caused By CI, Status Change, Operator Update
Open, Caused By CI, Closed, End
Open, Status Change, Assignment, End, Closed, Caused By CI, Operator Update
Open, End
Open, Closed, End, Caused By CI
Open, Reassignment, Assignment, End, Closed, Caused By CI, Status Change, Operator Update, Update
Open, Operator Update, Assignment, End, Closed, Caused By CI, Status Change
Open, Update, Assignment, End, Closed, Caused By CI, Status Change, Operator Update, Reassignment
Closed, Open, Assignment, End, Caused By CI, Status Change, Operator Update, Reassignment, Update
Closed, End
Closed, Caused By CI, End, Open, Assignment, Status Change, Operator Update, Reassignment, Update
Caused By CI, Closed, Open, Assignment, End, Status Change, Operator Update, Reassignment, Update
Caused By CI, End
Caused By CI, Open, Assignment, End, Closed, Status Change, Operator Update, Reassignment, Update
Status Change, Assignment, End, Closed, Open, Caused By CI, Reassignment, Update, Operator Update
Status Change, Closed, Open, Assignment, End, Caused By CI, Operator Update, Reassignment, Update
Status Change, Caused By CI, Closed, Open, Assignment, End, Operator Update, Reassignment, Update
Status Change, Operator Update, Assignment, End, Closed, Open, Caused By CI, Reassignment, Update
Assignment, End
Assignment, Closed, Open, Caused By CI, End, Status Change, Operator Update, Reassignment, Update
Assignment, Caused By CI, Closed, Open, Status Change, Operator Update, End, Reassignment, Update
Assignment, Status Change, Closed, Open, Caused By CI, End, Reassignment, Update, Operator Update
Assignment, Operator Update, Caused By CI, Closed, Open, Status Change, End, Reassignment, Update
Reassignment, Assignment, End, Closed, Open, Caused By CI, Status Change, Operator Update, Update
Reassignment, Update, Assignment, End, Closed, Open, Caused By CI, Status Change, Operator Update
Update, Assignment, End, Closed, Open, Caused By CI, Status Change, Operator Update, Reassignment
Update, Closed, Open, Assignment, End, Caused By CI, Status Change, Operator Update, Reassignment
Update, Reassignment, Assignment, End, Closed, Open, Caused By CI, Status Change, Operator Update
Operator Update, Assignment, End, Closed, Open, Caused By CI, Status Change, Reassignment, Update
Operator Update, Caused By CI, Closed, Open, Assignment, End, Status Change, Reassignment, Update
-----------------------------------------------------------------------------------------------------
```

Figure 20 – Output of aspects exclusively by cluster approach with 1% threshold

This real life events case study shows that the method can help specialist by suggesting aspects. All of the interviews yield different results. For example, the interviewed #1 and #3 found “Open” as aspect, while interviewed #2 did not. There are bigger differences, as interviewees #1 and #2 did not consider other elements as aspect beyond activities. Interviewee #3 considered the element gateway with the same decision taking as aspect. Another difference was the clustering of aspects none of the clustering was equal. For example, interviewee #1 clustered “Assignment”, “Reassignment” and “Update”, while interviewee #2 clustered “Update”, “Assignment” and “Status Change” and interviewee #3 clustered “Assignment”, “Reassignment”. These clusters are very similar, but different nonetheless. This emphasizes the subjective problem of human identification.

They all agree with the aspects found by the method can be actually an aspect. There were some aspects that the method did not find. It was the case of the identification of activities with similar meaning (*i.e.*, “Communication with Customer” and “Mail to Customer”). The next step for the method is to figure out how to add this functionality to improve the results.

However, there were aspects found by the method that were not found by the interviewees (i.e., “Start”, “End”). The interviewees agreed with these aspects and the justification for this gap is that they did not think to consider events. Besides the subjective problem, this is another advantage of the use of the method. Table 2 presents the comparison from the aspects found by the method with the aspects found by the interviewees.

The aspects found by the interviewee #1 and #3 are the elements which matches to the definition of aspect – modularization of crosscutting concern. They identified as crosscutting concerns elements repeated several times, elements used by different others elements, elements that can be reused in others domains, and elements that are independent of others elements. Accordingly to Table 2, the method could identify most of the aspects between these processes.

Besides the problems found and the threats listed to the validity of the method, there are others factors to consider as the method limitation. The tagging from the method is not always successful. There are some words that the library cannot tag, this results with words tagged null and the rest of the algorithm cannot find synonyms. This error can affect the method accuracy.

The factor that it also needs to be considered is the limitation of the topic of the processes loaded. The method finds aspects considering event existence in all loaded processes. Therefore, it is possible that are aspects in one process, which it is not present in the others processes, that it will not be identified. However, besides the limitations, threats and problems, the interviewee’s opinion is that the method can help specialists to find aspects.

6. Conclusion

Elements used in business process models can be scattered (repeated) within different processes, making it difficult to handle changes, analyze process for improvements, or check crosscutting impacts. These scattered elements are named Aspects. Similar to the aspect-oriented paradigm in programming languages, in BPM, aspect handling has the goal to modularize the crosscutting concerns spread across the models.

The use of aspects in models of business processes improves the process modularization, facilitates maintenance, understanding, modeling and reuse parts of the process. Therefore, it facilitates management of the process. However, it is a new field; so, the existing approaches of aspect-oriented models in BPM are being developed following different lines of work. In existing approaches, the identification is performed manually, resulting in the problem of subjectivity and lack of systematization.

In the literature, the identification of aspects in an aspect-oriented programming is most achieved by three approaches (Clone, Cluster and Fan-in analysis) [Barbosa, 2008]. The Clone approach aims to discover the methods in the code that can be clones of others, *i.e.*, duplicated code. The Cluster approach aims to group patterns of methods by their execution. The approach of this work is to identify aspects in business process model inspired in the first two software engineering methods (clones and clusters).

6.1. Contributions

This research proposed an approach for automatic identification of aspects from event logs (the process as it was executed – as-is – and not how it should have been executed – to-be). The method is based on mining techniques and it aims to solve the problem of the subjectivity identification made by specialists. Jalali [2014a] pointed out that the mining techniques to discover duplicated activities within one process are an ongoing research with many open questions. This dissertation developed a solution to fill this gap by discovering aspects among different processes. The method contribution is that it defined aspects from the event logs point of view based on software engineering techniques.

The method assumes at least two logs files as input. Another assumption is the log format, which it needs to be XES (standard format define by IEEE process mining task force [Van der Aalst *et al.,* 2011]). The last assumption is the level of abstractions between the processes are the same.

At first, the method reads the logs and extracts the process elements. Then, it applies the clone approach to identify aspect candidates considering elements with same names or synonym names. Afterwards, the third activity is to apply the cluster approach, which identifies aspect groups considering the order of execution of the aspects candidates identified by the previous step. The final activity is a manual action done by a specialist to decide from the list of aspects candidates provided by the method which aspects candidates is really an aspect.

The results from the proof of concept and real life logs presented in chapter 5 show that the proposed method can be used to assist specialist to identify aspects through mining the processes logs. Therefore, the results from the method are positive about aspects identification.

6.2. Limitations and threats

The first implementation of the method has limitations and threats. The first limitation is not being able to identify aspects from different level of abstraction. The second limitation is the process context. The aspects are found among the log files loaded. This feature makes the aspect identification dependent of context. The method finds aspects considering event existence in all loaded processes. It is possible that the aspect found from log #1 and log #2 is not present in log #3. Therefore, if loaded only log #1 and log #3, the aspect found between log#1 and log #2 will not be identified. Accordingly, this results that the validity of the study is dependent of the context from the logs loaded. The basic limitation of the method is that depends on events logs, *i.e.*, the method only can be used in processes which generate events logs.

There are also threats to the method. One threat is the manner the events names are written. The method uses natural language processing in its implementation. Therefore, the method takes in consideration the good practices to name activities in business process management. This good practice is the name is verb plus noun, verb plus object [MENDLING *et al.,* 2010]. Another threat is the tagging task. The tagging used in the method is not always successful. Sometimes, the library used cannot part-of-speech correctly or the library cannot tag the word. This result in words tagged as null, impairing the algorithm, leading in not finding synonyms. This error decreases the method accuracy. The basic threat is the quality of the events logs, *i.e.*, if the events logs load does not follow the good practices for names and structures, the method will not generate a good output.

6.3. Future Work

The solution developed is a technical method to identify aspects in business process management without depending on experts. The results are optimistic and for future work would be developed the analysis fan-in approach. Another future work is semantic analysis of the names of the activities reducing the complexity of synonyms algorithm and improving its specificity. It also calculates semantic similarity to find abstractions. That is, to identify activities that cover the meaning of others activities. Besides, there is the analysis of activities relations, such as successors and their predecessors of the activities as a way to prioritize the search for aspects.

Another future work is to include this method on process mining tools to provide access to it and to support the identification of aspects in business process management. This future work is the development of a ProM plugin. The plugin will be able to identify aspects from event logs loaded in the framework.

In additional of the future work of implementations, new case studies must be done with different logs (dataset) and domain experts to improve the method evaluation and to have future improvements features.

References

ARRAJ, V. 2010. "*ITIL®: the basics.*" Buckinghampshire, UK.

AZEVEDO, L.G.; SANTORO, F.; BAIAO, F.; SOUZA, J.; REVOREDO, K.; PEREIRA, V.; HERLAIN, I. 2009. "*A Method for Service Identification from Business Process Models in a SOA Approach*". In T. Halpin et al., eds. Enterprise, Business-Process and Information Systems Modeling. pp. 99-112.

AZEVEDO, L. G.; BAIAO, F.; SANTORO, F.; SOUZA, J. F. 2011. "*A Business Aware Service Identification and Analysis Approach*". In: IADIS International Conference Information Systems 2011, March, 11-13, Avila, Spain.

BARBOSA, F. S. "*Comparing Three Aspect Mining Techniques.*" 2008. Doctoral Symposium in Informatics Engineering (DSIE'08). Portugal.

BLOOMBERG, J.; SCHMELZER, R. 2006. "*Service Orient or Be Doomed!: How Service Orientation Will Change Your Business*". Hoboken, NJ: John Wiley & Sons.

BOSE, R. J. C.; VAN DER AALST, W. M. 2009. *"Abstractions in process mining: A taxonomy of patterns"*. In Business Process Management (pp. 159-175). Springer Berlin Heidelberg.

BUIJS, J. C. A. M.; VAN DONGEN, B. F.; VAN DER AALST, W. M. P. 2013. "*Mining Configurable Process Models from Collections of Event Logs*". BPM (Business Process Management) 11th Conference, China.

CAMPOS J. P.; BRAGA J. L.; RESENDE A. M. P.; SILVA C. H. O. 2010. "*Identification of aspect candidates by inspecting use cases descriptions*". SIGSOFT (Special Interest Group on Software Engineering), vol 35, pp 1-9.

CAPPELLI, C.; LEITE, J.; BATISTA, T.; SILVA, L. 2009. "*An Aspect-Oriented Approach to Business Process Modeling.*". EA-AOSD (15th Early Aspects Workshop - 8th International Conference on Aspect-Oriented Software Development), USA.

CAPPELLI, C.; SANTORO, F. M.; LEITE, J. C. S. P.; MEDEIROS, A. L.; BATISTA, T.; ROMEIRO, C. S. C. 2010. "*Reflections on the modularity of business process models. The case for introducing the aspect-oriented paradigm*". BPM Journal Vol. 16.

CASACHI, R. A.; CAMOLESI, A. R. 2012. "*Uso de Programação Orientada a Aspecto no Desenvolvimento de Aplicações que utilizam conceitos de Tecnologia Adaptativa*". Adaptive Technology Workshop.

CHARFI, A.; MEZINI, M. 2004. "*Aspect-oriented web service composition with AO4BPEL*". In Web Services (pp. 168-182). Springer Berlin Heidelberg.

CHARFI A.; MÜLLER H.; MEZINI M. 2010. “*Aspect-Oriented Business Process Modeling with AO4BPMN*”. In T. K. et al., editor, Modelling Foundations and Applications, volume 6138 of LNCS, pages 48-61. Springer.

COLLELL, D. C. 2012. “*Aspect-oriented modeling of business processes*”. Master's thesis, der Technischen Universitat Darmstadt, Darmstadt.

DAVENPORT, T. 1994. “*Reengenharia de processos*”. Rio de Janeiro: Campus, p.6-8.

DUMAS, M.; LA ROSA, M.; MENDLING, J.; REIJERS, H. A. 2013. “*Fundamentals of Business Process Management*”. Springer.

FINLAYSON, M. A. 2014. “*Java Libraries for Accessing the Princeton Wordnet: Comparison and Evaluation.*” Proceedings of the 7th Global Wordnet Conference. Tartu, Estonia.

GARCIA, R. 2010. "*O que é Programação Orientada a Aspectos?*". Java Framework Portal. Available at <http://www.javaframework.org/portal/2010/04/14/o-que-programao-orientada-a-aspectos>. Accessed in May, 2014.

GÜNTHER, W. C.; ROZINAT, A. 2012. *“Disco: Discover Your Processes*”. BPM (Demos), v. 940, p. 40-44.

GÜNTHER, W. C.; VERBEEK E. 2014. "*OpenXES - Developer Guide 2.0*". Available at: http://www.xes-standard.org/openxes/developerguide. Accessed on February, 2015.

GÜNTHER, W. C.; VAN DER AALST, W. M. 2007. *“Fuzzy mining–adaptive process simplification based on multi-perspective metrics”.* In Business Process Management (pp. 328-343). Springer Berlin Heidelberg.

JALALI, A. 2014. "*Assessing aspect oriented approaches in business process management.*" Perspectives in Business Informatics Research. Springer International Publishing, p231-245. (a)

JALALI, A. 2014. "*Aspect Mining in Business Process Management.*" Perspectives in Business Informatics Research. Springer International Publishing, p246-260. (b)

JOSUTTIS, N. 2007. “*SOA in practice: The Art of Distributed System Design*”. Beijing; Cambridge. O’Reilly, 324p.

KICZALES, G.; HUGUNIN, J.; HILSDALE, E.; KERSTEN, M.; PALM, J.; LOPES, C.; GRISWOLD, B.; ISBERG, W. 2003. "*Aspect Oriented Programming*". Final Technical Report from Air Force Research Laboratory from Palo Alto Research Center. Rome, New York. July.

KICZALES, G.; LAMPING, J.; MENDHEKAR, A.; MAEDA, C.; LOPES C.; LOINGTIER J.; IRWIN, J. 1997. “*Aspect-Oriented Programming*”. European Conference on Object-Oriented Programming (ECOOP), Finland. Springer-Verlag LNCS 1241. June.

MENDLING, J.; REIJERS, H. A.; RECKER, J. 2010. "*Activity labeling in process modeling: Empirical insights and recommendations.*" Information Systems, v. 35, n. 4, p. 467-482.

OLIVEIRA, S. B. 2006. *"Gestão pro Processos: fundamentos, técnicas e modelos de implementação"*. Rio de Janeiro: Qualitymark. page 291-305.

OMG. 2011. "*Business Process Modeling Notation Specification*". January.

OSTROFF, F. 1999. *"The Horizontal Organization: what the organization of the future actually looks like and how it delivers value to customers"*. Page 257. USA: Oxford University Press.

PAHLSSON, N. 2012. "*Aspect-Oriented Programming - An Introduction to Aspect-Oriented Programming and AspectJ*". Department of Technology from University of Kalmar. Topic Report for Software Engineering. Novembro.

RAGO, A.; ABAIT E.; MARCOS C.; DIAZ-PACE A. 2009. "*Early Aspect Identification From Use Cases Using Nlp And Wsd Techniques*". 15th Workshop on Early aspects from International Conference on Aspect-Oriented Software Development.

RECKER, J. 2012. *"Scientific research in information systems: a beginner's guide"*. Springer Science & Business Media.

REICHERT, M.; HALLERBACH, A.; BAUER, T. 2015. *"Lifecycle management of business process variants"*. In: Handbook on Business Process Management 1. Springer Berlin Heidelberg, p. 251-278.

RICHETTI, P. H. P.; BAIÃO F. A.; SANTORO F. M. 2014. "*Declarative Process Mining: Reducing Discovered Models Complexity by Pre-Processing Event Logs.*" Business Process Management. Springer International Publishing, p400-407.

ROZINAT, A.; DE MEDEIROS, A. K. A.; GÜNTHER, C. W.; WEIJTERS, A. J. M. M.; VAN DER AALST, W. M. 2008. "*The need for a process mining evaluation framework in research and practice*". In Business Process Management Workshops (pp. 84-89). Springer Berlin Heidelberg. January.

RUBIN, V.; LOMAZOVA, I.; VAN DER AALST, W. M. P. 2014. *"Agile development with software process mining"*. In: Proceedings of the 2014 International Conference on Software and System Process. ACM, 2014. p. 70-74.

SAMPAIO A.; CHITCHYAN R.; RASHID A.; RAYSON P. 2005. "*EA-Miner: a tool for automating aspect-oriented requirements identification*". 20th IEEE/ACM International Conference on Automated software engineering.

SANTOS, F. J. N.; LEITE, J. C. S. P.; CAPPELLI, C.; BATISTA, T. V.; SANTORO, F. M. 2011. "*Using goals to identify aspects in business process models.*" Proceedings of the 2011 international workshop on Early aspects. ACM.

SHARP, A.; MCDERMOTT, P. 2001. *"Workflow Modeling: Tools for Process Improvement and Application Development"*. Artech House - ISBN 1-58053-021-4

SIGNAVIO. 2014. "*BPM Academic Initiative*". Available at < http://www.signavio.com/bpm-academic-initiative/>. Accessed in December, 2014.

SILVA, L.F. 2006. "*An aspect-oriented strategy for requirements modeling*", PhD thesis, Computer Science Department, PUC-Rio, Rio de Janeiro, March.

SOARES, A. H. V.; ROCHA, A. DE R.; ALVES, F. L.; ALVES, J. C. 2012. "*Programação Orientada a aspectos - uma visão geral*". Department of Computer Science of the Federal University of Lavras.

SOUZA A.; CAPPELLI C.; SANTORO F.; AZEVEDO L. G.; LEITE J. C. S. DO P. 2011. "*Service Identification in Aspect-Oriented Business Process Models*". SOSE (Service Oriented System Engineering) 6th International Symposium.

TAVARES F.; MARINHO L. 2014. "*AO-BPM 2.0: Modelagem de Processos Orientada a Aspectos*". Final graduation work. Available at: http://bsi.uniriotec.br/tcc/publicacoes.html.

TOUTANOVA K; KLEIN D; MANNING C; SINGER Y. 2003. "*Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network.*" In Proceedings of HLT-NAACL 2003, pp. 252-259.

VALLE, R.; OLIVEIRA, S. B. 2012. *"Análise e Modelagem de Processos de Negócio. Foco na Notação BPMN"*. São Paulo: Atlas, 2012.

VAN DER AALST, W. M. P. 2013. *"Process cubes: Slicing, dicing, rolling up and drilling down event data for process mining"*. In Asia Pacific Business Process Management (pp. 1-22). Springer International Publishing.

VAN DER AALST, W. M. P.; WEIJTERS A. J. M. M. 2004. "*Process mining: a research agenda*". Journal Computers in Industry – Special Issue: Process/Workflow mining Vol. 53 Issue 3, pp. 231 – 244. Elsevier Science Publishers B. V. Amsterdam. Abril.

VAN DER AALST, W.M.P.; ADRIANSYAH, A.; ALVES DE MEDEIROS, A.K.; ARCIERI, F.; BAIER, T.; BLICKLE, T.; BOSE, J.C.; BRAND, P.C.W. VAN DEN, BRANDTJEN, R.; BUIJS, J.C.A.M.; BURATTIN, A.; CARMONA, J.; CASTELLANOS, M.; CLAES, J.; COOK, J.; COSTANTINI, N.; CURBERA, F.; DAMIANI, E.; LEONI, M. DE; DELIAS, P.; DONGEN, B.F. VAN; DUMAS, M.; DUSTDAR, S.; FAHLAND, D.; FERREIRA, D.R.; GAALOUL, W.; GEFFEN, F. VAN; GOEL, S.; GUNTHER, C.W.; GUZZO, A.; HARMON, P.; HOFSTEDE, A.H.M. TER; HOOGLAND, J.; INGVALDSEN, J.E.; KATO, K.; KUHN, R.; KUMAR, A.; LA ROSA, M.; MAGGI, F.M.; MALERBA, D.; MANS, R.S.; MANUEL, A.; MCCREESH, M.; MELLO, P.; MENDLING, J.; MONTALI, M.; MOTAHARI NEZHAD, H.; MUEHLEN, M. ZUR; MUNOZ-GAMA, J.; PONTIERI, L.; RIBEIRO, J.T.S.; ROZINAT, A.; SEGUEL PERÉZ, H.; SEGUEL PÉREZ, R.E.; SEPÚLVEDA, M.; SINUR, J.; SOFFER, P.; SONG, M.S.; SPERDUTI, A.; STILO, G.; STOEL, C.; SWENSON, K.; TALAMO, M.; TAN, W.; TURNER, C.; VANTHIENEN, J.; VARVARESSOS, G.; VERBEEK, H.M.W.; VERDONK, M.C.; VIGO, R.; WANG, J.; WEBER, B.; WEIDLICH, M.; WEIJTERS, A.J.M.M.; WEN, L.; WESTERGAARD, M.; WYNN, M.T. 2011. "*Process mining manifesto*". In F. Daniel, K. Barkaoui & S.

Dustdar (Eds.), Business Process Management Workshops (BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part I), (Lecture Notes in Business Information Processing, 99, pp. 169-194). Berlin: Springer.

VAN DONGEN, B. F.; DE MEDEIROS, A. K. A.; VERBEEK, H. M. W.; WEIJTERS, A. J. M. M.; VAN DER AALST, W. M. 2005. *"The ProM framework: A new era in process mining tool support"*. In Applications and Theory of Petri Nets 2005 (pp. 444-454). Springer Berlin Heidelberg.

VERBEEK, H.M.W. 2010. *"ProM 6 Getting Started*". ProM documentation. Available at < http://www.promtools.org/prom6/downloads/prom-6.0-getting-started.pdf >. September.

WANG, J.; ZHU, J.; LIANG, H.; XU, K. 2005. "*Aspect-Oriented Business Process Modeling*". Research Report, IBM – China.

WESKE, M. 2012. "*Business process management: concepts, languages, architectures.*" Springer Science & Business Media.

WfMC. 1999. *"Workflow management coalition terminolog and glossary"*. Technical Report WfMC-TC-1011, Workflow Management Coalition, Brussels. Februrary,1999.

Appendix I. **Real Life Events Processes**

This section presents the process models for services components “WBS000091” (Figure 21), “WBS000263” (Figure 22) and “WBS000073” (Figure 23).

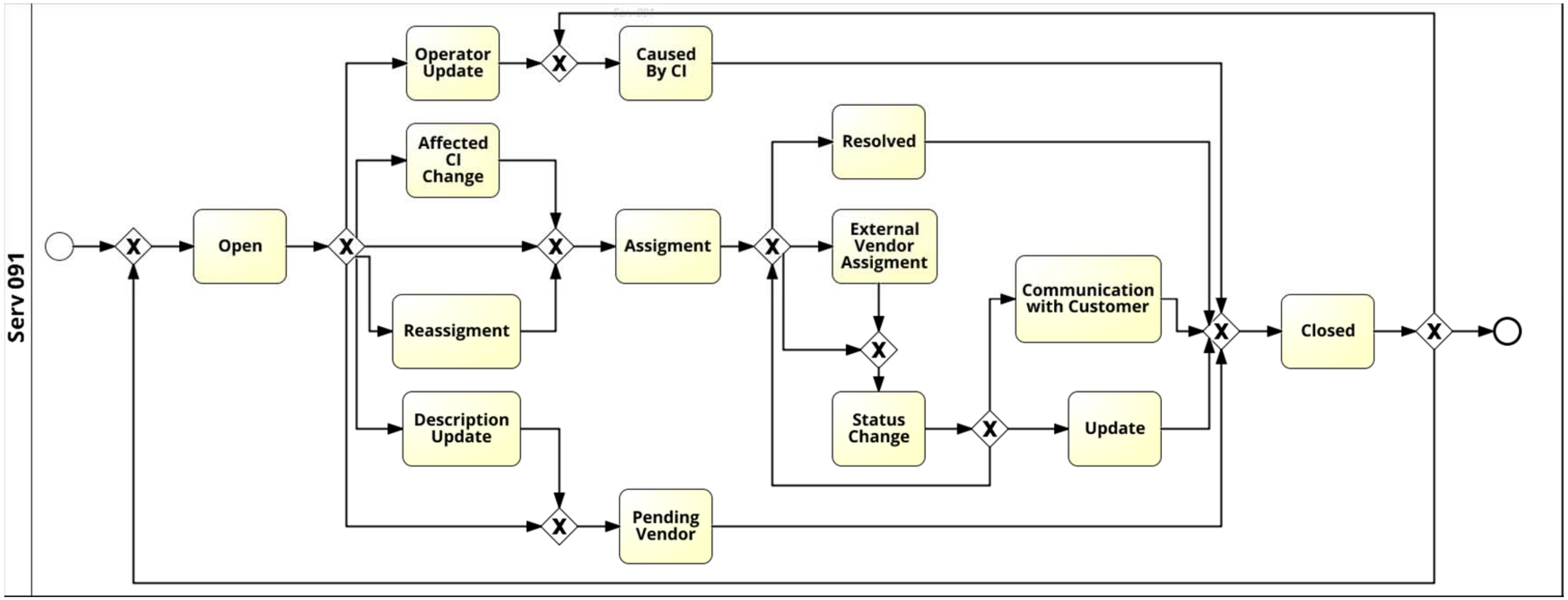


Figure 21 – Process model of Service Component “WBS000091”.

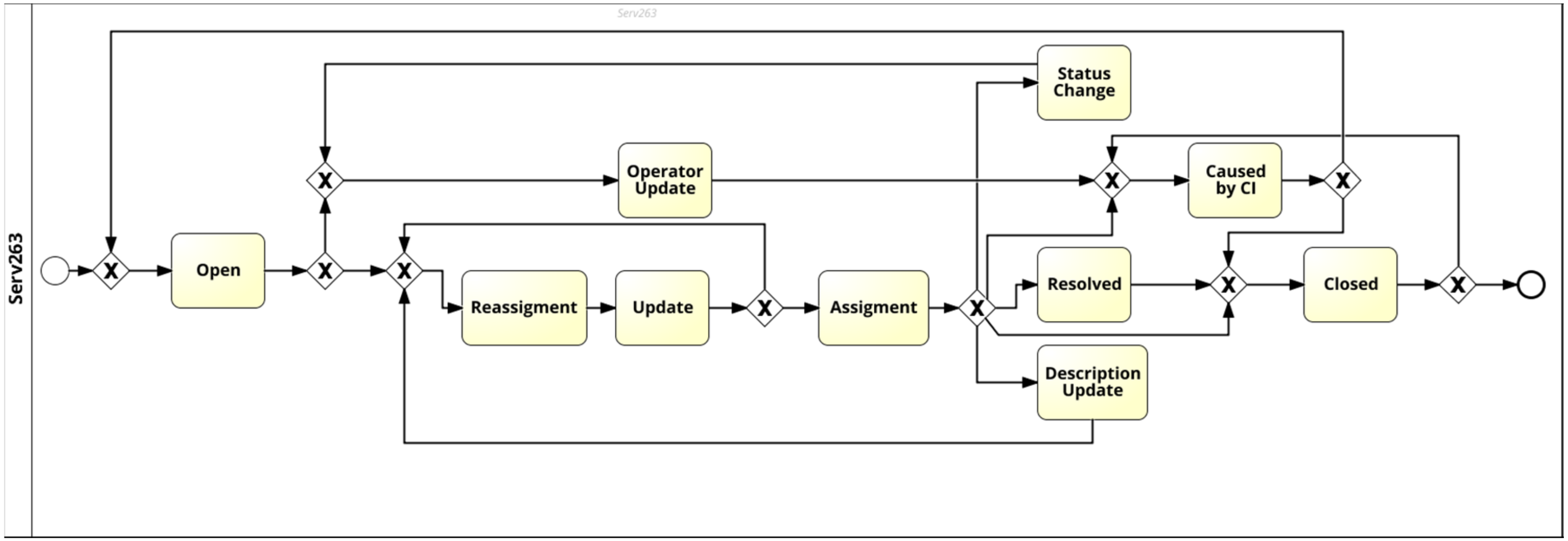


Figure 22 – Process model for Service Component “WBS000263”.

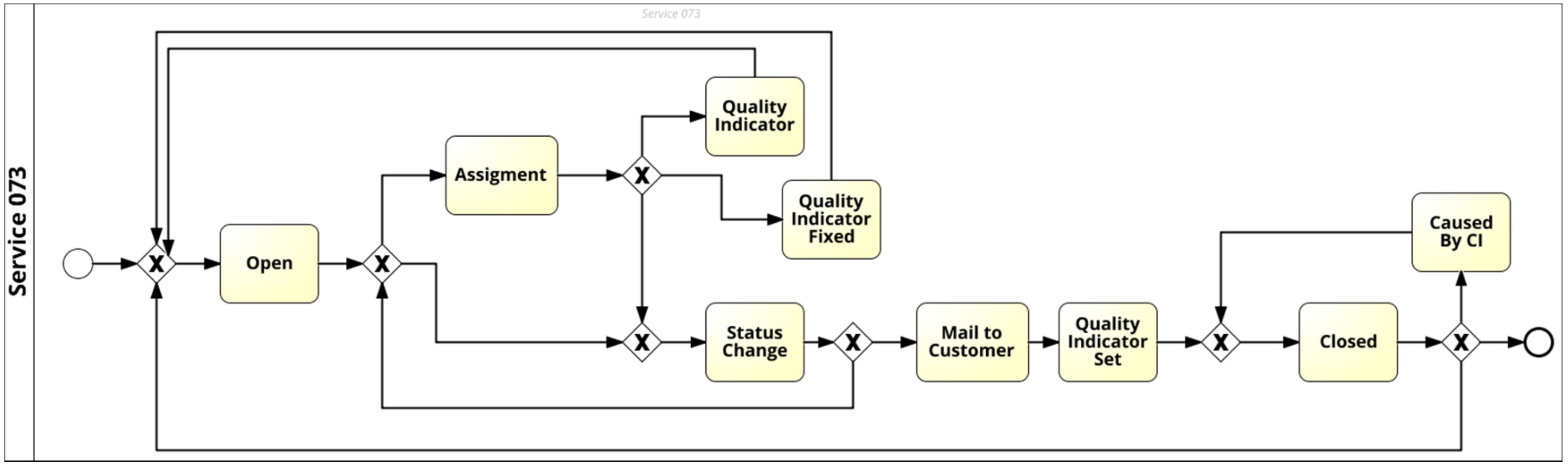


Figure 23 – Process model for Service Component “WBS000073”

Appendix II. **Results Comparison**

This section presents the table comparing the results between the method and the interviewees' identification.

Table 2 - Aspect results comparison.

	Start End	Open	Closed	Caused By CI	Status Change	Assignment	Reassignment	Update	Operator Update	Description Update	Resolved	Mail to Customer	Communication with Customer	Gateway before Mail to Customer	Gateway after closed
Method	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	X	X	X	X
Interviewee #1	X	✓	✓	X	✓	✓	✓	✓	✓	✓	X	✓	✓	X	X
Interviewee #2	X	X	✓	X	✓	✓	✓	✓	X	X	X	X	X	X	X
Interviewee #3	X	✓	✓	X	✓	✓	✓	✓	✓	✓	X	✓	✓	✓	✓

Appendix III. **Log Files**

Beginning of the synthetic log file of “Require Discipline Program Process” with only one trace.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- XES version 1.0 -->
<!-- Created by Fluxicon Disco (http://fluxicon.com/disco/ -->
<!-- (c) 2012 Fluxicon Process Laboratories - http://fluxicon.com/ -->
<log xes.version="1.0" xmlns="http://www.xes-standard.org" xes.creator="Fluxicon Disco">
  <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
  <extension name="Lifecycle" prefix="lifecycle" uri="http://www.xes-standard.org/lifecycle.xesext"/>
  <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
  <extension name="Organizational" prefix="org" uri="http://www.xes-standard.org/org.xesext"/>
  <global scope="trace">
    <string key="concept:name" value="name"/>
  </global>
  <global scope="event">
    <string key="concept:name" value="name"/>
    <string key="lifecycle:transition" value="transition"/>
    <string key="org:resource" value="resource"/>
    <date key="time:timestamp" value="2015-03-21T18:12:53.266-03:00"/>
    <string key="Activity" value="string"/>
    <string key="Resource" value="string"/>
  </global>
  <classifier name="Activity" keys="Activity"/>
  <classifier name="Resource" keys="Resource"/>
  <string key="lifecycle:model" value="standard"/>
```

```
<string key="creator" value="Fluxicon Disco"/>
<string key="library" value="Fluxicon Octane"/>
<trace>
  <string key="concept:name" value="61"/>
  <string key="creator" value="Fluxicon Disco"/>
  <event>
    <string key="concept:name" value="Discipline program  required"/>
    <string key="lifecycle:transition" value="complete"/>
    <string key="org:resource" value="NOT_SET"/>
    <date key="time:timestamp" value="2015-03-10T12:17:12.654-03:00"/>
    <string key="Activity" value="Discipline program  required"/>
  </event>
  <event>
    <string key="concept:name" value="Fill form"/>
    <string key="lifecycle:transition" value="complete"/>
    <string key="org:resource" value="NOT_SET"/>
    <date key="time:timestamp" value="2015-03-10T12:26:32.478-03:00"/>
    <string key="Activity" value="Fill form"/>
  </event>
  <event>
    <string key="concept:name" value="Print discipline program"/>
    <string key="lifecycle:transition" value="complete"/>
    <string key="org:resource" value="NOT_SET"/>
    <date key="time:timestamp" value="2015-03-10T14:33:09.510-03:00"/>
    <string key="Activity" value="Print discipline program"/>
  </event>
  <event>
    <string key="concept:name" value="Stamp all files"/>
    <string key="lifecycle:transition" value="complete"/>
    <string key="org:resource" value="NOT_SET"/>
    <date key="time:timestamp" value="2015-03-11T11:09:38.013-03:00"/>
    <string key="Activity" value="Stamp all files"/>
  </event>
  <event>
    <string key="concept:name" value="Sign all files"/>
    <string key="lifecycle:transition" value="complete"/>
    <string key="org:resource" value="NOT_SET"/>
    <date key="time:timestamp" value="2015-03-11T14:56:58.907-03:00"/>
    <string key="Activity" value="Sign all files"/>
  </event>
  <event>
    <string key="concept:name" value="File form"/>
    <string key="lifecycle:transition" value="complete"/>
    <string key="org:resource" value="NOT_SET"/>
    <date key="time:timestamp" value="2015-03-12T12:45:38.231-03:00"/>
    <string key="Activity" value="File form"/>
  </event>
  <event>
    <string key="concept:name" value="Discipline program  obtained"/>
    <string key="lifecycle:transition" value="complete"/>
    <string key="org:resource" value="NOT_SET"/>
    <date key="time:timestamp" value="2015-03-12T12:45:38.231-03:00"/>
    <string key="Activity" value="Discipline program  obtained"/>
  </event>
</trace>
```

Beginning of the real life events log file of Service 091 with only one trace.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- XES version 1.0 -->
<!-- Created by Fluxicon Disco (http://fluxicon.com/disco/ -->
<!-- (c) 2012 Fluxicon Process Laboratories - http://fluxicon.com/ -->
<log xes.version="1.0" xmlns="http://www.xes-standard.org" xes.creator="Fluxicon Disco">
  <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
  <extension name="Lifecycle" prefix="lifecycle" uri="http://www.xes-standard.org/lifecycle.xesext"/>
  <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
  <extension name="Organizational" prefix="org" uri="http://www.xes-standard.org/org.xesext"/>
  <global scope="trace">
    <string key="concept:name" value="name"/>
  </global>
  <global scope="event">
    <string key="concept:name" value="name"/>
    <string key="lifecycle:transition" value="transition"/>
    <string key="org:resource" value="resource"/>
    <date key="time:timestamp" value="2015-03-16T01:10:51.823-03:00"/>
    <string key="INCIDENTACTIVITY_TYPE" value="string"/>
    <string key="ASSIGNMENT_GROUP" value="string"/>
  </global>
  <classifier name="Activity" keys="INCIDENTACTIVITY_TYPE"/>
  <classifier name="Resource" keys="ASSIGNMENT_GROUP"/>
  <string key="lifecycle:model" value="standard"/>
  <string key="creator" value="Fluxicon Disco"/>
  <string key="library" value="Fluxicon Octane"/>
  <trace>
    <string key="concept:name" value="IM0001274"/>
    <string key="creator" value="Fluxicon Disco"/>
    <event>
      <string key="concept:name" value="Start"/>
      <string key="lifecycle:transition" value="complete"/>
      <string key="org:resource" value="Start"/>
      <date key="time:timestamp" value="2013-01-10T08:48:00.000-02:00"/>
    <string key="INCIDENTACTIVITY_TYPE" value="Start"/>
    <string key="ASSIGNMENT_GROUP" value="Start"/>
  </event>
  <event>
    <string key="concept:name" value="Open"/>
    <string key="lifecycle:transition" value="complete"/>
    <string key="org:resource" value="TEAM0008"/>
    <date key="time:timestamp" value="2013-01-10T08:48:00.000-02:00"/>
    <string key="INCIDENTACTIVITY_TYPE" value="Open"/>
    <string key="ASSIGNMENT_GROUP" value="TEAM0008"/>
  </event>
  <event>
    <string key="concept:name" value="Closed"/>
    <string key="lifecycle:transition" value="complete"/>
    <string key="org:resource" value="TEAM0075"/>
    <date key="time:timestamp" value="2013-01-10T09:14:00.000-02:00"/>
    <string key="INCIDENTACTIVITY_TYPE" value="Closed"/>
    <string key="ASSIGNMENT_GROUP" value="TEAM0075"/>
  </event>
  <event>
    <string key="concept:name" value="Caused By CI"/>
    <string key="lifecycle:transition" value="complete"/>
    <string key="org:resource" value="TEAM0075"/>
    <date key="time:timestamp" value="2013-01-10T09:14:00.000-02:00"/>
    <string key="INCIDENTACTIVITY_TYPE" value="Caused By CI"/>
    <string key="ASSIGNMENT_GROUP" value="TEAM0075"/>
  </event>
  <event>
    <string key="concept:name" value="End"/>
    <string key="lifecycle:transition" value="complete"/>
    <string key="org:resource" value="End"/>
    <date key="time:timestamp" value="2013-01-10T09:14:00.000-02:00"/>
    <string key="INCIDENTACTIVITY_TYPE" value="End"/>
    <string key="ASSIGNMENT_GROUP" value="End"/>
  </event>
</trace>
```

Appendix IV. **Original Log**

This section presents a snapshot of the original log files. Table 3 is the snapshot from the interaction dataset. Table 4 is from the incident dataset and Table 5 is from the incident activity dataset.

Table 3 - Interaction dataset.

CI Name (aff)	CI Type (aff)	CI Subtype (aff)	Service Comp WBS (aff)	Interaction ID	Status	Impact	Urgency	Priority	Category	KM number	Open Time (First Touch)	Close Time	Closure Code	First Call Resolution	Handle Time (secs)	Related Incident
SBA000243	application	Server Based Application	WBS000125	SD0000001	Closed	5	4	4	incident	KM0000987	09/09/2011 09:23	14/02/2014 09:05	Other	N	239	IM0000001
SUB000443	subapplication	Web Based Application	WBS000125	SD0000002	Closed	4	4	4	request for information	KM0000989	29/09/2011 14:59	13/12/2013 16:27	Software	N	406	IM0000001
LAP000110	computer	Laptop	WBS000187	SD0000003	Closed	4	4	4	incident	KM0000317	13/10/2011 15:47	21/10/2013 05:01	Software	N	738	
DTA000110	application	Desktop Application	WBS000256	SD0000004	Closed	4	4	4	incident	KM0000057	01/12/2011 15:39	21/10/2013 05:02	Unknown	N	787	
SBA000855	application	Server Based Application	WBS000054	SD0000005	Closed	4	4	4	incident	KM0000652	23/12/2011 16:23	21/10/2013 05:02	Software	N	459	IM0000003
SUB000424	subapplication	Web Based Application	WBS000073	SD0000006	Closed	4	4	4	incident	KM0000702	16/01/2012 14:09	21/10/2013 05:03	Other	N	412	
SUB000508	subapplication	Web Based Application	WBS000162	SD0000007	Closed	4	4	4	incident	KM0000553	05/02/2012 13:26	04/11/2013 13:51	Other	N	363	IM0000004
DTA000616	application	Desktop Application	WBS000092	SD0000008	Closed	3	3	3	incident	KM0000988	09/02/2012 12:38	21/10/2013 05:03	Software	N	374	
CBD000079	computer	Banking Device	WBS000147	SD0000009	Closed	2	2	2	incident	KM0000132	15/02/2012 15:46	21/10/2013 05:04	Hardware	N	272	
SBA000207	application	Server Based Application	WBS000123	SD0000010	Closed	3	3	3	incident	KM0000488	27/02/2012 07:52	17/01/2014 10:41	Other	N	295	

Table 4 - Incident dataset.

CI Name (aff)	CI Type (aff)	CI Subtype (aff)	Service Component WBS (aff)	Incident ID	Status	Impact	Urgency	Priority	Category	KM number	Alert Status	# Reassignments	Open Time	Reopen Time	Resolved Time	Close Time	Handle Time (Hours)	Closure Code	# Related Interactions	Related Interaction	# Related Incidents	# Related Changes	Related Change	CI Name (CBy)	CI Type (CBy)	CI Subtype (CBy)	ServiceComp WBS (CBy)
SUB000508	subapplication	Web Based Application	WBS000162	IM0000004	Closed	4	4	4	incident	KM0000553	closed	26	05/02/2012 13:32		04/11/2013 13:50	04/11/2013 13:51	3871,691	Other	1	SD0000007	2			SUB000508	subapplication	Web Based Application	WBS000162
WBA000124	application	Web Based Application	WBS000088	IM0000005	Closed	3	3	3	incident	KM0000611	closed	33	12/03/2012 15:44	02/12/2013 12:31	02/12/2013 12:36	02/12/2013 12:36	4354,786	Software	1	SD0000011	1			WBA000124	application	Web Based Application	WBS000088
DTA000024	application	Desktop Application	WBS000092	IM0000006	Closed	3	3	3	request for information	KM0000339	closed	3	29/03/2012 12:36		13/01/2014 15:12	13/01/2014 15:13	4843,119	No error - works as designed	1	SD0000017				DTA000024	application	Desktop Application	WBS000092
WBA000124	application	Web Based Application	WBS000088	IM0000011	Closed	4	4	4	incident	KM0000611	closed	13	17/07/2012 11:49		14/11/2013 09:31	14/11/2013 09:31	43,21833	Operator error	1	SD0000025				WBA000124	application	Web Based Application	WBS000088
WBA000124	application	Web Based Application	WBS000088	IM0000012	Closed	4	4	4	incident	KM0000611	closed	2	10/08/2012 11:01		08/11/2013 13:55	08/11/2013 13:55	3383,903	Other	1	SD0000029				SUB000508	subapplication	Web Based Application	WBS000162
WBA000124	application	Web Based Application	WBS000088	IM0000013	Closed	4	4	4	incident	KM0000611	closed	4	10/08/2012 11:27		08/11/2013 13:54	08/11/2013 13:54	3383,437	Other	1	SD0000031				SUB000508	subapplication	Web Based Application	WBS000162

WBA000082	application	Web Based Application	WBS000055	IM0000014	Closed	4	4	4	incident	KM0000401	closed	2	15/08/2012 14:17		27/12/2013 10:59	27/12/2013 10:59	3703,191	Unknown	1	SD0000033			WBA000082	application	Web Based Application	WBS000055
WBA000124	application	Web Based Application	WBS000088	IM0000015	Closed	4	4	4	incident	KM0000611	closed	5	22/08/2012 16:31		08/11/2013 14:09	08/11/2013 14:09	3294,624	Other	1	SD0000034			WBA000124	application	Web Based Application	WBS000088
WBA000124	application	Web Based Application	WBS000088	IM0000017	Closed	3	3	3	incident	KM0000611	closed	2	29/08/2012 15:59		08/11/2013 14:02	08/11/2013 14:02	0,862778	Other	1	SD0000036			#N/B	#N/B	#N/B	#N/B
WBA000082	application	Web Based Application	WBS000055	IM0000018	Closed	4	4	4	incident	KM0000401	closed	2	03/09/2012 16:04		08/11/2013 14:33	08/11/2013 14:35	3211,527	No error - works as designed	1	SD0000037			#N/B	#N/B	#N/B	#N/B
WBA000124	application	Web Based Application	WBS000088	IM0000019	Closed	4	4	4	incident	KM0000611	closed	6	21/09/2012 12:56		08/11/2013 14:23	08/11/2013 14:23	3067,449	Software	1	SD0000040	1	C00000056	#N/B	#N/B	#N/B	#N/B
WBA000124	application	Web Based Application	WBS000088	IM0000020	Closed	4	4	4	incident	KM0000611	closed	8	01/10/2012 10:49		08/11/2013 14:18	08/11/2013 14:22	1322,619	Software	1	SD0000042			#N/B	#N/B	#N/B	#N/B
WBA000082	application	Web Based Application	WBS000055	IM0000021	Closed	4	4	4	incident	KM0000401	closed	5	02/10/2012 12:12	28/01/2014 14:07	04/02/2014 09:38	04/02/2014 09:38	1132,428	Software	2	#MULTIVALUE			WBA000082	application	Web Based Application	WBS000055

Table 5 - Incident activity dataset.

Incident ID	DateStamp	IncidentActivity_Number	IncidentActivity_Name	Assignment Group	KM number	Interaction ID
IM0000004	07/01/2013 08:17	001A3689763	Reassignment	TEAM0001	KM0000553	SD0000007
IM0000004	04/11/2013 13:41	001A5852941	Reassignment	TEAM0002	KM0000553	SD0000007
IM0000004	04/11/2013 13:41	001A5852943	Update from customer	TEAM0002	KM0000553	SD0000007
IM0000004	04/11/2013 12:09	001A5849980	Operator Update	TEAM0003	KM0000553	SD0000007
IM0000004	04/11/2013 12:09	001A5849979	Assignment	TEAM0003	KM0000553	SD0000007
IM0000004	04/11/2013 13:41	001A5852942	Assignment	TEAM0002	KM0000553	SD0000007
IM0000004	04/11/2013 13:51	001A5852172	Closed	TEAM0003	KM0000553	SD0000007
IM0000004	04/11/2013 13:51	001A5852173	Caused By CI	TEAM0003	KM0000553	SD0000007
IM0000004	04/11/2013 12:09	001A5849978	Reassignment	TEAM0003	KM0000553	SD0000007
IM0000004	25/09/2013 08:27	001A5544096	Operator Update	TEAM0003	KM0000553	SD0000007
IM0000005	03/06/2013 11:15	001A4725475	Update	TEAM9999	KM0000611	SD0000011
IM0000005	03/04/2013 11:29	001A4327777	Operator Update	TEAM0003	KM0000611	SD0000011
IM0000005	07/01/2013 08:17	001A3689771	Reassignment	TEAM0001	KM0000611	SD0000011
IM0000005	05/09/2013 08:58	001A5377163	Operator Update	TEAM0003	KM0000611	SD0000011
IM0000005	12/04/2013 11:03	001A4396943	Operator Update	TEAM0003	KM0000611	SD0000011
IM0000005	23/04/2013 08:22	001A4466088	Status Change	TEAM0003	KM0000611	SD0000011
IM0000005	02/12/2013 12:00	001A6068111	Update from customer	TEAM0002	KM0000611	SD0000011
IM0000005	02/12/2013 12:32	001A6068174	Reassignment	TEAM0002	KM0000611	SD0000011
IM0000005	02/12/2013 12:32	001A6068175	Assignment	TEAM0002	KM0000611	SD0000011
IM0000005	02/12/2013 12:36	001A6068564	Caused By CI	TEAM0003	KM0000611	SD0000011
IM0000005	02/12/2013 12:36	001A6068563	Closed	TEAM0003	KM0000611	SD0000011
IM0000005	02/12/2013 12:32	001A6068177	Update from customer	TEAM0002	KM0000611	SD0000011
IM0000005	02/12/2013 12:32	001A6068176	Status Change	TEAM0002	KM0000611	SD0000011
IM0000005	27/03/2013 08:57	001A4287716	Operator Update	TEAM0003	KM0000611	SD0000011
IM0000005	16/04/2013 14:19	001A4419476	Status Change	TEAM9999	KM0000611	SD0000011

IM0000005	12/04/2013 11:03	001A4396942	Status Change	TEAM0003	KM0000611	SD0000011
IM0000005	24/05/2013 08:42	001A4664412	Reassignment	TEAM9999	KM0000611	SD0000011
IM0000005	02/12/2013 11:48	001A6065225	Reassignment	TEAM0003	KM0000611	SD0000011
IM0000005	02/12/2013 11:48	001A6065226	Assignment	TEAM0003	KM0000611	SD0000011
IM0000005	02/12/2013 11:48	001A6065227	Operator Update	TEAM0003	KM0000611	SD0000011
IM0000005	27/03/2013 08:57	001A4287717	Description Update	TEAM0003	KM0000611	SD0000011
IM0000005	03/04/2013 11:29	001A4327776	Assignment	TEAM0003	KM0000611	SD0000011
IM0000005	03/06/2013 11:14	001A4725473	Assignment	TEAM9999	KM0000611	SD0000011
IM0000005	03/04/2013 11:29	001A4327775	Reassignment	TEAM0003	KM0000611	SD0000011
IM0000005	16/04/2013 14:19	001A4419475	Assignment	TEAM9999	KM0000611	SD0000011
IM0000005	24/05/2013 08:42	001A4664414	Analysis/Research	TEAM9999	KM0000611	SD0000011
IM0000005	24/05/2013 08:42	001A4664413	Assignment	TEAM9999	KM0000611	SD0000011
IM0000005	03/06/2013 11:15	001A4725474	Assignment	TEAM9999	KM0000611	SD0000011
IM0000011	14/11/2013 09:31	001A5926549	Closed	TEAM0004	KM0000611	SD0000025